Logistical Planning of Mobile Food Retailers Operating Within

Urban Food Desert Environments

by

Christopher Wishon

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved October 2016 by the
Graduate Supervisory Committee:

ARIZONA STATE UNIVERSITY

December 2016

ABSTRACT

Mobile healthy food retailers are a novel alleviation technique to address disparities in access to urban produce stores in food desert communities. Such retailers, which tend to exclusively stock produce items, have become significantly more popular in the past decade, but many are unable to achieve economic sustainability. Therefore, when local and federal grants and scholarships are no longer available for a mobile food retailer, they must stop operating which poses serious health risks to consumers who rely on their services.

To address these issues, a framework was established in this dissertation to aid mobile food retailers with reaching economic sustainability by addressing two key operational decisions. The first decision was the stocked product mix of the mobile retailer. In this problem, it was assumed that mobile retailers want to balance the health, consumer cost, and retailer profitability of their product mix. The second investigated decision was the scheduling and routing plan of the mobile retailer. In this problem, it was assumed that mobile retailers operate similarly to traditional distribution vehicles with the exception that their customers are willing to travel between service locations so long as they are in close proximity.

For each of these problems, multiple formulations were developed which address many of the nuances for most existing mobile food retailers. For each problem, a combination of exact and heuristic solution procedures were developed with many utilizing software independent methodologies as it was assumed that mobile retailers would not have access to advanced computational software. Extensive computational tests were performed on these algorithm with the findings demonstrating the advantages of the developed procedures over other algorithms and commercial software.

The applicability of these techniques to mobile food retailers was demonstrated through a case study on a local Phoenix, AZ mobile retailer. Both the product mix and routing of the retailer were evaluated using the developed tools under a variety of conditions and assumptions. The results from this study clearly demonstrate that improved decision making can result in improved profits and longitudinal sustainability for the Phoenix mobile food retailer and similar entities.

ACKNOWLEDGMENTS

I am incredibly thankful to my fiancé Julianne for her patience and commitment throughout my graduate education. I couldn't have survived this process without her and I am especially glad she agreed to extend her stay in the desert on my behalf. I couldn't have fallen in love with a better person. I also have to thank my parents for their support and open arms for the last six years. I am so thankful that they were always there to listen and guide me whenever I got lost and I hope I can repay their kindness for years to come.

I would like to thank Dr. J. René Villalobos for all of his support and advice throughout my doctoral research. He provided the perfect combination of guidance and freedom allowing me to push my boundaries while still providing a crucial safety net during my early years. His feedback and open door policy were essential to the completion of my research and the experiences he provided that extended beyond my dissertation were crucial to my development as a scholar. I must also thank Dr. Christopher Wharton and Dr. Pitu Mirchandani for serving on my committee and providing critical advice and feedback to improve my research. I would also like to especially thank Dr. John Fowler for serving on my committee and his trust and support to allow me to serve as an instructor for the past three years. His guidance and advice have been essential to advancing my teaching abilities and the opportunities he provided were crucial to sparking my interest in an academic career.

A large thank you also goes to all my friends who supported me throughout this process and made the past six years more bearable. A special thanks goes to Hector Flores, Patrick James, Corey Balint, Sarah Conte, Jonathan Adler, Nicholas Mason, and Phillip Howard.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

One of the greatest health challenges currently facing the U.S. is the increasing rates of diet-related diseases and issues. Since 1980, obesity, as measured by the body mass index (BMI), has increased by more than 15 percentage points with no statistically significant decreases in the rate of obesity over that time period (Flegal et al. 2012). As of 2010, this increase in BMI has resulted in the U.S. having the highest estimated prevalence of overweight adult men and women, defined as a BMI greater than or equal to 25, among the top economically developed 50 countries. Recent research has also demonstrated that only 40% of adults in the U.S. maintain a healthy weight, defined as a BMI between 18.5 and 25.0, and only 23% consume the recommended 5 servings of fruits and vegetables (Reeves and Rafferty 2005). Further complicating this issue is that the rate of food insecurity has drastically increased in recent years. As of 2012, low or very-low food insecurity rates for households, defined as not having or unable to obtain enough food for all members of the household due to insufficient resources, has risen to 14.5% of all U.S. household which represents over a 6 million household increase from 2000 (Coleman-Jensen et al. 2012).

These diet related diseases and issues have significant negative outcomes both at the individual and population level. As of 2005, it was estimated 17% of all preventable deaths can be attributed to dietary risk factors which include but is not limited to high dietary trans fatty acids, high dietary salts, low intake of fruits and vegetables, and low dietary omega-3 fatty acids (Danaei et al. 2009). Furthermore, it was shown that obesity, which is not directly attributable to dietary risk factors, accounts for an additional 11% of the preventable deaths in 2005. While this inability to maintain a healthy diet and weight is generally only considered to have effects at the individual level, the cumulative effect of the inability for U.S. citizens to maintain a healthy diet has drastically increased medical expenditures, decreased productivity, and reduced potential length of life. Specifically, by analyzing five of the health conditions for which diet is a major risk factor (coronary heart disease, cancer, stroke, diabetes, and osteoporotic hip fractures), it was estimated that consuming a healthier diet would save the U.S. population $114.5 billion per year in 2012 as measured by decreasing healthcare expenditures and increasing productivity and lifespan (Anekwe and Rahkovsky 2013). This evidence demonstrates that a high percentage of Americans are unable to maintain a healthy diet and this inability has severe repercussions at both the individual and societal level.

While diets are generally conceived to be based on an individual's choices, recent evidence has shown that there may be external factors which influence a person's ability to maintain a healthy diet and weight. One such influence is the local food environment. Generally, the local food environment for an individual is characterized by the quantity and type of food retailers or establishment (hereafter referred to collectively as food providers) in the surrounding areas around that person's place of residence. However, this environment can also be expanded to include the food options around a person's workplace as well as along frequently traveled transit routes. The concept of the local food environment has only recently become the focus of academic research with results demonstrating that the type and quantity of food providers is correlated with obesity (Robert and Reither 2004; Lopez 2007; Rundle et al. 2007), food intake (Diez Roux et al. 1999), physical activity (Nelson et al. 2006), and overall health (Stimpson et al. 2007).

Coinciding with this increased focus on the local food environment by academic researchers, government agencies have also started to recognize that the quantity and assortment of food providers around an individual’s work and residence can influence that person’s health. To address this issue, the United States Department of Agriculture's (USDA) 2008 farm bill defined a “food desert” as an “area in the United States with limited access to affordable and nutritious food, particularly such an area composed of predominantly lower income neighborhoods and communities” (*Food, Conservation, and Energy Act of 2008* 2008). As part of the legislation passed in the 2008 farm bill, the USDA developed a set of criteria which specify which communities are food deserts, thus allowing priority to be given to food access projects in those areas. As is common with most food desert research, the USDA uses U.S. census tracts as proxies for neighborhoods. By definition, census tracts are generally contiguous, statistical subdivisions of counties which follow visible or identifiable features such that the tract has a population between 1,200 and 8,000 people.

Furthermore, the USDA, as well as the majority of other research efforts, uses access to a traditional supermarket as a proxy for access to a nutritional food provider. The purpose is that most food desert studies do not verify the existence of healthy food options at the stores, but use supermarkets as they are most likely to have the broadest mix of healthy food options (Larson, Story, and Nelson 2009). Hence, the USDA uses supermarkets, supercenters, and large grocery stores to determine the availability of food retailers for food desert residents (Ver Ploeg and Dutko 2013).

Given these considerations, the USDA identifies a food desert if it meets both of the following criteria (Ver Ploeg and Dutko 2013):

- Low-Income Census Tract
  - Tract's poverty rate is greater than or equal to 20%, or
  - Tract's median family income is 80% or less of the state's median family income, or
  - An urban tract's median family income is 80% or less of the metropolitan's median family income
- Low-Access Census Tract
  - For an urban census tract, at least 500 people or 33% of the tract's population live more than 1 mile from the nearest supermarket
  - For a rural census tract, at least 500 people or 33% of the tract's population live more than 10 miles from the nearest supermarket

Using these criteria, the USDA have identified that there are nearly 9,000 urban and rural food desert census tracts which represent 12.3% of all nation-wide census tracts. Furthermore, urban census tracts, tracts whose population-weighted centroids are located in an area with more than 2,500 people (Ver Ploeg and Dutko 2013), comprise a majority of U.S. food desert tracts as over 7,500 urban census tracts qualify as food deserts which represents 13.6% of all urban U.S. tracts. These urban census tracts also tend to favor those with higher populations as there are nearly 33 million people living in urban food deserts which represents 13.8% of all urban residents. This provides clear evidence that food access is a systemic problem in many urban U.S. communities and therefore improving access to healthy and affordable food providers should be a priority within the afflicted neighborhoods.

One proposed methodology to alleviate food desert conditions within an afflicted urban community is to introduce new food retailers within that community (Rose et al. 2009). However, research has demonstrated that the number of supermarkets per resident purchasing power is the same within a food desert community as within a non-afflicted community (Alwitt and Donley 1997). Hence, a successful food desert intervention must be able to aggregate demand within a community using a different approach than traditional brick-and-mortar stores. One such intervention is to use a mobile retailer. This mobile retailer is typically a large vehicle (e.g. a repurposed bus or a large trailer pulled by a truck) which is stocked with healthy food items that are sold within food desert communities at specific locations according to a predetermined schedule. Even though this retail format for healthy foods did not exist 15 years ago, examples of these retailers can currently be found in over a dozen US cities as shown in Figure 1.

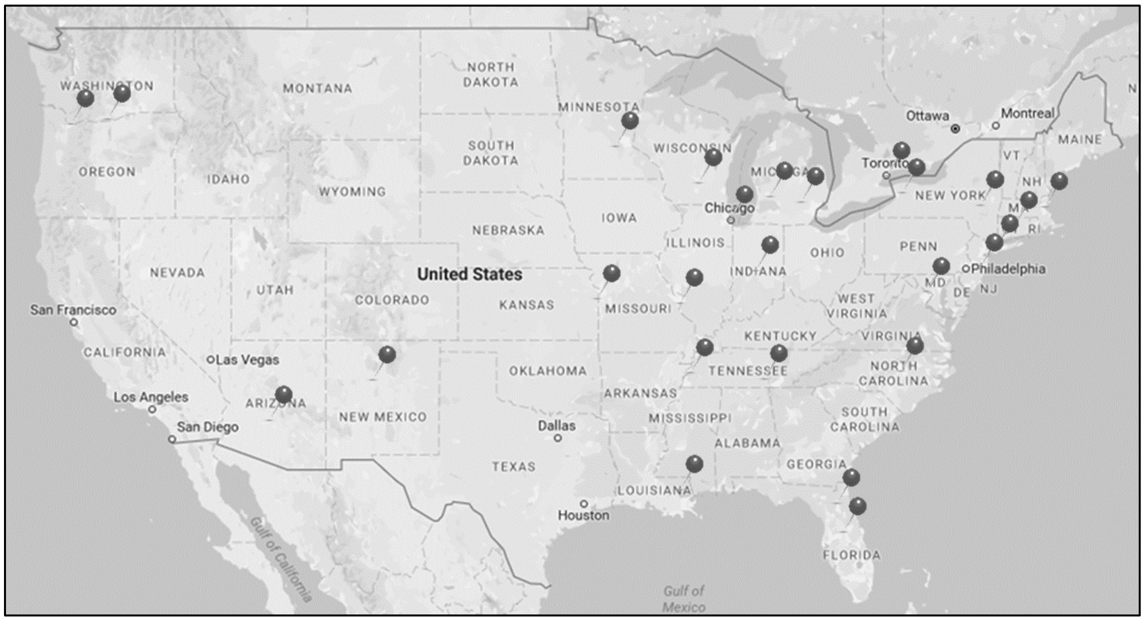


Figure 1. Map of current and former mobile healthy food retailers

In spite of the growing popularity of this retail format, mobile retailers have yet to demonstrate that they can be an effective alleviation technique since they have had minimal success at becoming economically sustainable. Most, if not all, existing mobile retailers receive a significant portion of their funding from federal or local grants and even long serving mobile retailers experience difficulties in obtaining sufficient income to offset their operational costs. Such a strategy is clearly unsustainable for any permanent alleviation remedy.

This initial inability for mobile retailers to reach economic sustainability has created significant concerns regarding the efficacy and longevity of the mobile retail format in alleviating food desert conditions. To address these concerns, this dissertation research was conducted to study the operational decision making capabilities of such a retailer. Specifically, mathematical models were developed which mimic the operational decisions faced by mobile fresh food retailers. Given these tools, the optimal operational decisions can be determined for a community and the results can be analyzed to determine the revenue potential for these types of retailers in their service neighborhood.

Two specific operational decisions will be investigated in this dissertation: the retailer product mix decision and the retailer routing/scheduling decision. The motivation for investigating these decisions, the key considerations for each decision, and the nuances of mobile food retailers are provided in the remaining sections of this chapter. The remainder of the dissertation is split into six main chapters. Chapter 2 will discuss all relevant literature including the motivation for addressing disparities in food access as well as the theoretical literature which will serve as the basis for the developed mathematical models. Chapter 3 will introduce the mathematical model and foundation for solving the product mix problem based on a general retailer which may have any number of constraints and requirements. Chapter 4 will continue this discussion by providing a specific, high quality method for determining the optimal product mix for those retailers whose only requirement is that the stocked product meets a given revenue threshold. Chapter 5 introduces the retailer routing and scheduling problem and provides a solution methodology to solve the problem optimally. Chapter 6 continues this discussion by providing faster but approximate techniques to determine a routing plan. Within each of the aforementioned chapters, computational experiments are performed for each of the developed solution methodologies and the results from these experiments will be discussed in the appropriate chapter. Chapter 7 introduces and provides the results from a detailed case study using the developed decision making tools, along with an additional routing tool which can incorporate time window constraints, using data from a Phoenix, AZ fresh food retailer.

1.1 Mobile food retailer background

Prior to discussing mobile food retailers in detail, it should be noted that other interventions exist which seek to address the disparities in food access in urban environments. The purpose of this section is to provide a cursory discussion of these techniques and to discuss the strengths and weaknesses of each in order to motivate the use of mobile food retailers.

One of the most popular and self-evident alleviation strategies is to introduce a new, traditional food retailer into the community. Since food desert communities lack such retailers by definition, introducing a new retailer should remedy the food desert conditions. This approach has been promoted as a possible solution by nearly all major food desert studies including the main study from the USDA (Ver Ploeg et al. 2009). This approach is further supported by market forces as numerous food store chains have been created which are specifically designed to satisfy low-income demand as demonstrated by Save-A-Lot, ALDI, and Food-4-Less (Ver Ploeg et al. 2009). Further support for this strategy is that introducing supermarkets into underserved communities has received significant national and local funding. For instance, numerous funding opportunities exist at the federal level including New Market Tax Credits, Community Development Block Grants, and the Empowerment Zone Program (Ver Ploeg et al. 2009).

Even with this support, there is limited research demonstrating the societal impact from introducing new, traditional food retailers as only two studies have been performed which documented the longitudinal effects of such interventions. The first research was by Wrigley, Warm, and Margetts (2003) whose principal observation was that residents who had poor diets prior to the retailers introduction increased their fruit and vegetable intake by a half portion while those with the worst diets increased their intake by a full portion on average. The second research study was by Cummins et al. (2005) whose major finding is that there was no statistical evidence that the new supermarket increased consumption of healthy foods or improved physiological health but they did find that the community self-reported improved psychological health.

This research, albeit limited, does indicate that introducing a supermarket is not guaranteed to be an effect alleviation strategy. This is further supported by supplemental research such as that performed by Alwitt and Donley (1997) who found that supermarkets in Chicago, IL, a city in which disparities in access are typically found, are evenly distributed with respect to purchasing power within the community. Hence, new supermarkets may be infeasible as there is insufficient demand for them to remain within the community. Further supporting this finding is the Low Supermarket Access (LSA) areas identified by The Reinvestment Fund (TRF) (Califano et al. 2012). TRF noted that over half of the LSA areas do not have the sufficient income to support a new food retailer. The study by Rose et al. also support this observation as they state, "all neighborhoods cannot support a supermarket, nor are supermarkets the only way to assure access to healthy food." (Rose et al. 2009).

In addition to introducing new food retailers, one of the other alleviation methodologies which has become popular is to improve the supply or incentivize the purchasing of healthy and affordable foods within non-traditional food retailers. Examples of such retailers include convenience stores, gas stations, and local corner stores. This strategy is supported by numerous sources including the USDA (Ver Ploeg et al. 2009) and Rose et al. (2009) and is supported by federal programs such as the Healthy Bodega Initiative which incentivizes bodegas in New York City to increase the quantity of low-fat milk, fruits, and vegetables offered in their stores. While some research has investigated this strategy, such as Weatherspoon et al. (2013) who found that food desert residents in Detroit, MI increased their purchases of healthy food items when such items were made more available, there are clear barriers to this alleviation strategy. First and foremost, the fixed and operational costs of installing freezers and coolers to supply fresh and healthy foods are often too costly for smaller stores. These stores also frequently sell unhealthy foods and increasing the demand for healthy food in such stores may increase the cross-selling of unhealthy food items. Finally, such an approach is not feasible if these retailers are not already prevalent within the afflicted community.

The final alleviation strategy which is common within communities with food access disparities is community-driven solutions such as introducing farmers' markets and/or community gardens or facilitating access to existing food providers by improving the urban infrastructure or public transport. Examples of these strategies include incentivizing Supplemental Nutrition Assistance Program (SNAP) purchases at farmers' markets within Arizona which increased both SNAP and general purchases (Bertmann et al. 2012) and offering discounted bus passes for SNAP members in Madison, WI to facilitate access to existing food retailers (Ver Ploeg et al. 2009). While these approaches have shown tremendous success within certain communities, there still remain significant barriers. Principally, low-income shoppers perceive farmers' market and similar efforts as higher priced shopping alternatives compared with supermarkets (Balsam, Webber, and Oehlke 1994; Zepeda 2009; Flamm 2011) even though such a difference typically does not exist (Flamm 2011; McGuirt et al. 2011). This is further complicated if the market does not accept nutritional assistance vouches as low-income shoppers frequently cite this issue as a common barrier (Flamm 2011; Leone et al. 2012). Additionally, many low-income shoppers state they have poor physical access to local markets (Grace et al. 2007; Racine, Smith Vaughn, and Laditka 2010; Ruelas et al. 2012) or do not even know a market exists in their areas (Racine, Smith Vaughn, and Laditka 2010; Leone et al. 2012).

This research demonstrates that all of the standard alleviation strategies face significant challenges when attempting to alleviate food desert conditions within an urban community. For instance, the community may not have the requisite purchasing power to support a traditional retailer while existing farmers' markets may not be a viable option as they may not support SNAP purchases or food desert residents may believe the produce items are not priced competitively. In comparison, a mobile food retailer naturally addresses many of these issues, such as being able to aggregate dispersed demand, indicating it could be a successful food desert alleviation strategy. However, existing mobile retailers have experienced significant challenges when attempting to become economically sustainable as most, if not all, existing mobile retailers receive a significant portion of their funding from federal or local grants and many have ceased operations due to a lack of funds. Such a strategy is clearly unsustainable which indicates research into mobile retailers is necessary to determine if they can become lasting intervention within the communities they serve.

To best study mobile food retailers, the decision making capabilities of the retailers will be investigated for two reasons. First and foremost, developing mathematical models which simulate the decision making of mobile retailers will allow for robust tests to be conducted which simulate mobile food retailer profits under a variety of community conditions and assumptions. Secondly, the developed mathematical models can be provided to mobile retailers to assist them with their decision making capabilities.

In particular, two decision making epochs were identified as the most significant to current and future mobile retailers: the product mix problem and the vehicle routing/scheduling problem. These operational level decisions were identified as they are the decisions which are the most within the control of current mobile retailer. Other decisions such as the strategic level distribution, operating region, and warehouse selection problems and the tactical level vehicle selection and vehicle quantity decisions are important, but are not fully within the control of most modern mobile retailers. For instance, most retailers arise as a result of a direct need within a given community which implies the retailer cannot independently select their operating region. The following subsections detail both the product mix and vehicle routing problems to motivate the literature review in Chapter 2 and the developed mathematical models and solution methods detailed in the remaining chapters.

1.2 Mobile food retailer product mix

The first operational decision is to determine the product mix which is stocked on the mobile retailer. To date, the most common approach for stocking healthy mobile retailers is to exclusively sell fresh fruits and vegetables. The advantage of this approach is that these items are always considered to be healthy and they are generally the foods which are lacking in food desert communities. However, some mobile retailers decide to offer a more diverse array of foods including fruits, vegetables, low-fat milks, and whole grain breads. This mix requires more space and infrastructure, but the diverse offerings more closely match the product types found in traditional food retailers. Hence, a robust modeling methodology is needed to encompass all of these possibilities.

In particular, the developed mathematical model must be able to consider a wide variety of produce and grocery items. Given these food items, one of the goals for most mobile food retailers is to select a product mix such that the overall nutritional value of the mix is as high as possible or meets certain criteria. This is especially important if the mobile food retailer wishes to balance healthy and unhealthy food items in order to increase the earning potential of the mix. Similarly, the model must consider the overall consumer price of the mix as many of the retailers serve low-income populations. Finally, the model must allow the retailer to specify either a required profit margin for the product mix or allow the retailer to require the profit margin to serve as the objective which should be maximized. Given these goals, the literature review of possible modelling approaches will be provided in Chapter 2 while Chapters 3 and 4 discuss two modeling approaches and solution methodologies which are able to address all possible product mix needs for any mobile food retailer type.

1.3 Mobile food retailer scheduling and routing

The other operational decision for mobile food retailers is the locations and times when the retailer stops within their community. In most cases, the retail stops are locations in which it is presumed that low-income, underserved people visit with relative frequency at the time of service. However, it is unclear which of these location should be selected, if more options are provided than can possibly be serviced, and in which order they should be visited. In addition, it is possible that some of the retailer’s customers would be willing to travel to a nearby visited location if the retailer does not directly travel to that customer’s ideal location. Typically, the distance a customer would be willing to travel would have to be small, but such customer travel is plausible in an urban setting.

Hence, the mathematical model must incorporate numerous considerations to make the developed solutions applicable to mobile food retailers. Specifically, the model must be able to develop feasible vehicle routes which stop and start at a central depot, only visit a subset of locations within certain time frames, and not attempt to visit more locations than it can possible service given its stocked product mix in one trip. In addition, an ideal solution methodology would also factor in the ability for customer travel between locations as visiting centralized customer locations may yield higher quality routes. Given these objective, the literature review of feasible vehicle routing modelling approaches will be provided in Chapter 2 while Chapters 5 and 6 discuss multiple modeling approaches and solution methodologies based on the balance of solution quality and speed desired by the mobile food retailer. An additional model is also provided in Chapter 7 which is specifically designed for the retailer which serves as the basis for the Phoenix, AZ case study.

1.4 Conclusion

In summary, one of the factors which has been identified as driving diet-related and nutrition-related diseases is an individual's local food environment. To remedy disparities in the food environment, numerous strategies have been recommended or trialed which include adding new full-service traditional retailers, improving existing small non-traditional retailers (e.g. corner stores), improving low-income access to farmers' markets and equivalent community driven initiatives, and improving urban infrastructure. In addition to these techniques, one novel strategy has been to introduce mobile food retailers which sell fresh fruits and vegetables by traveling through communities while stopping at participating locations according to a predetermined schedule. The advantage of this format is mobile retailers are able to aggregate dispersed demand as it is assumed these communities cannot support traditional full-service supermarkets. The disadvantage of this format is that many of the existing mobile retailers have stopped their operations as they were unable to become economically sustainable. Instead, they were able to operate as long as they were continually supported through grants and sponsorships. Such a strategy is clearly unsustainable and the goal of this dissertation is to develop techniques to improve the economic sustainability of the mobile retail format.

Two key issues were identified which are within the control of most modern mobile food retailers while also having the greatest impact on their economic sustainability: the mobile food retailer product mix decision and the mobile food retailer routing and scheduling problem. For the product mix decision, the goal of the mobile retailer is to stock a healthy, profitable mix which also has a low cost for their customers. For the routing and scheduling decision, the goal of the mobile retailer is to efficiently serve as many low-income, food desert customers as possible while minimizing the cost of routing. For the remainder of this dissertation, it will be assumed that all low-income, food desert customers are equivalent and therefore the goal of the routing and scheduling decision is to earn a maximum profit which is a function of customer revenue less transportation costs.

The goal of this dissertation is to address these two key issues by developing mathematical formulations which model the product mix and routing decisions. With these models, the current performance of mobile retailers can be measured against the optimal decisions thereby allowing conclusions regarding the economic sustainability of mobile food retailers. Additionally, these models can be developed such that they can utilized by mobile food retailers to aid in their logistical and operational planning.

Parts of this introduction chapter were included in Wishon and Villalobos (2016a).

CHAPTER 2

LITERATURE REVIEW

This chapter addresses the existing research and literature related to the operational decisions of mobile healthy food retailers which service urban, underserved neighborhoods and communities. First, the motivations for addressing the operational decisions of mobile healthy food retailers will be discussed with specific emphasis on the extent and effect of food deserts. These motivations were briefly summarized in Chapter 1, but the breadth of literature on food deserts requires a separate and more thorough analysis. Following this summary, the potential modeling and solution approaches for the product mix and vehicle routing operational decisions will be discussed. Within each section, the chosen model and solution approach will be provided which reflects the algorithms and analyses which are detailed in the remaining chapters.

2.1 Extent and effect of food deserts

Considerable effort has been undertaken to determine the prevalence of food desert communities by researchers across numerous disciplines and organizations. Since summarizing all of this research is outside of the scope of this discussion, the following section presents the most recent research into *statistically significant* disparities in food access. For readers interested in further food desert research, summaries of existing literature are provided by Caspi et al. (2012), Walker, Keane, and Burke (2010), McKinnon et al. (2009), and Beaulac, Kristjansson, and Cummins (2009).

The key issue with current research into the existence of food deserts is that there is little consistency when defining the community, healthy food, and access criteria required for an area to be classified as a food desert. For instance, Ver Ploeg et al. (2009), the study accompanying the USDA's definition of a food desert, used 1 km. grids as proxies for communities while defining poor access to healthy foods as living more than a half mile (or full mile) away from the nearest supermarket with additional requirements including the resident being low-income and possibly not having access to personal transportation. In comparison, Rose et al. (2009) used census tracts to define a community and measured disparity based on access to all food retailer types within a 1 (or 2) km. radius while considering the actual shelf space dedicated to fruits and vegetables in these stores. While it is not the focus of this dissertation, this important shortcoming is mentioned because these differences in parameters are pervasive throughout food desert literature and hinder a concise conclusion on the existence and extent of food deserts. Readers interested in a thorough discussion of this and other issues within food desert literature are recommended to read Bitler and Haider (2011).

Given this shortcoming, existing food desert studies are still useful in identifying local statistical disparities in access. For instance, numerous studies into the existence of food deserts indicate that at-risk demographic groups have statistically less access to supermarkets then their reference groups. This applies to low-income versus high-income (Baker et al. 2006; Block and Kouba 2007; Larsen and Gilliland 2008), Black versus non-Hispanic White (Baker et al. 2006; Bader et al. 2010; Powell, Slater, et al. 2007), and Hispanic versus non-Hispanic White (Moore and Diez Roux 2006; Bader et al. 2010). However, the opposite is true for access to smaller or independent food retailers since it appears that most at-risk groups have better access to these types of stores (Morland et al. 2002; Moore and Diez Roux 2006; Powell, Slater, et al. 2007). For the purposes of this summary, smaller stores include independent grocery stores, fruit and vegetable markets, meat markets, farmers' markets, or similar independent food retailers.

These results demonstrate that even if urban, at-risk groups may be underserved by traditional retailers, smaller retailers could fill this gap. For instance, when the proportion of Mexican-Americans started to increase in the southwest US, small ethnic grocery stores called Carnicerias naturally arose as a method to address the need for community-specific foods (Duran 2007). Hence, it appears that natural economic development favors smaller food retailers within the urban food desert environment. Such a result is not surprising, but it is often overlooked by community developers seeking to alleviate food desert conditions and supports the use of mobile food retailers as an alleviation strategy.

Given these disparities, significant research has been undertaken to quantify the potential effects of having poor access to healthy foods. With respect to supermarket access, numerous studies have investigated if having poor access to supermarkets implies that at-risk communities pay more for their healthy food purchases. These studies have unanimously found that low-income citizens do not spend more on food items. In many instances, low-income urban populations have statistically less food expenditures than high-income shoppers even though studies have demonstrated that the stores more frequently located nearby low-income populations tend to have higher food prices (Kaufman et al. 1997; Andreyeva et al. 2008).

One theory for this phenomenon is that lower-income shoppers are more likely to rely on lower quality food items as measured by the look and freshness of the food. In support of this theory, Block and Kouba (2007) and Andreyeva et al. (2008) found that low-income shoppers have worse access to high-quality fruits and vegetables as opposed to high-income shoppers. Another factor is that low-income shoppers are extremely price sensitive with respect to food purchases (Caraher et al. 1998; Ohls et al. 1999; Clifton 2004; Alkon et al. 2013; Haynes-Maslow et al. 2013) and will therefore be more likely to travel further for bigger cost savings. Both of these theories demonstrate that any intervention must be very cost-conscious if the goal is to serve low-income communities.

Besides the financial effects, research has been conducted on the health implications of living within a food desert community. Numerous studies have identified that having statistically better access to supermarkets is either positively associated or has no effect on healthy eating habits (Rose and Richards 2004; Zenk et al. 2009) or on health outcomes such as obesity and disease (Morland, Wing, and Diez Roux 2002; Zenk et al. 2009; Bodor et al. 2010). While these results are not definitive, no statistically-based study was identified which indicated better access was associated with decreases in healthy food consumption or health outcomes.

Likewise, better access to smaller food retailers, applying the same definition used for identifying disparities in access, showed the same trend for diet quality/consumption (Lopez 2007; Gustafson et al. 2013) but had mixed correlations with obesity. Three out of the five studies which investigated the correlation between small store access and obesity found no relationship (Morland, Wing, and Diez Roux 2002; Bodor et al. 2008; Zenk et al. 2009) while two others found that better access was correlated with increased obesity (Powell, Auld, et al. 2007; Bodor et al. 2010). While this could be a causal relationship, these results can also be explained within the context of the prior findings since low-income and minority populations (especially Black Americans) tend to have better access to smaller food retailers and these groups are much more likely to be obese (Flegal et al. 2012).

It should also be noted that most of the studies investigating the effects of food deserts suffer from the same inconsistencies as the research which study the existence of disparities since there is no standard definition of a food desert. Therefore, comparing the results between two studies is challenging since different measures may have been employed. This disparity also prohibits precise, global measurements on the effects of food deserts since the definition differences between studies imply that the scale of the effects is incomparable.

In summary, the results on the extent and effect of food deserts is far from conclusive but numerous studies were able to identify statistically significant disparities in access across multiple communities and no studies identified that an increase in access to fresh and healthy foods had negative health impacts. This evidence demonstrates that seeking to address identified disparities in access is a justifiable use of local and federal funding and resources. However, it is recommended that researchers seeking to identify disparities in access and the effect of such disparities develop standard definitions so inter-community measurements can be conducted in future research.

This literature also indicates some of the key considerations which must be included when developing a mobile food retailer. For instance, research into food deserts have identified that low-income residents do not pay more for food on average (Kaufman et al. 1997; Andreyeva et al. 2008) even though their neighborhoods can have less availability. This demonstrates that low-income consumers are cost-conscious and will not buy expensive grocery items even if they are healthier. In addition, this literature also demonstrates the unhealthy food items may outsell healthy food items in smaller food stores thereby creating a rise in overall obesity. Hence, a mobile food retailer must limit the availability of such goods or completely eliminate them if they wish to positively affect the communities they serve.

2.2 Mobile food retailer product mix literature

Four separate approaches were considered as potential modeling techniques to address the mobile food retailer product mix problem: the supermarket product mix problem, healthy meal plan selection, the forward/reserve warehousing problem, and the knapsack problem. The key research from each of these areas is discussed in subsections 2.2.1 through 2.2.3. Subsection 2.2.3 concludes with a discussion of the final modeling approach and methodology using in this dissertation in Chapters 3 and 4.

2.2.1 Supermarket product mix and healthy meal plan literature

Research into how supermarkets determine their product mix and allocate space to their selected products share clear similarities with the mobile food retailer product mix problem as they both are constrained by floorplan availability, they both desire to offer a diverse product mix, and they both wish to stock grocery items to ensure an adequate profit margin. Literature into this topic include item substitution policies from Grashof (1970) and Heeler, Kearney, and Mehaffey (1973) who developed methodologies to determine the ideal item to aid to an existing product mix based on the attributes of the candidate items and the available shelf space. Such research eventually became more advanced as pseudo-knapsack problem models were developed which were able to stock an entire shelf/store given a list of candidate items. Examples of such research include

Zufryden (1986) whose models considered minimum stocking limits and geometry considerations and were solved via dynamic programming, Reyes and Frazier (2007) whose models considered item demand and were solved via goal programming, and Hansen and Heisbroek (1979) whose models considered shelving space and replenishment costs and were solved via Lagrangian relaxation.

While these models neither factor in the cost of the items for customer nor the health of the stocked food items, the model developed by the USDA for the Thrifty Food Plan (TFP) incorporates both (Carlson et al. 2007). Specifically, the TFP is based on a knapsack optimization model whose goal is to determine a minimum cost food plan which meets all dietary requirements and needs while not significantly deviating from customer preferences. The dietary requirements are incorporated as additional constraints in the optimization model and they include limits or requirements on carbohydrates, fiber, sodium, calories, protein, calcium, etc. The USDA uses this model to determine the lowest cost food plan which serves as the basis for federal SNAP reimbursement levels.

2.2.2 Forward and reserve warehouse allocation literature

Another research area which has potential applications to determining the mobile retailer product mix is the forward and reserve warehouse allocation problem. The forward and reserve allocation problem is to determine which and how many of each stock keeping unit (SKU) within a warehouse should be allocated to the forward storage area and the reserve storage area. The forward area, located closer to the central picker location, is the storage area for highly requested SKUs such that they are readily available and require little retrieval time. The reserve area stores less popular SKUs and generally in higher volumes (Bartholdi III and Hackman 2011). This is similar to the mobile retailer product mix problem since the supermarket can be modeled as the reserve area while the mobile retailer is the forward area. Popular food items, as measured by demand and health measures, are stocked in the mobile retailer which increases convenience and minimizes the picking time while less popular food items are only sold in traditional food retailers.

The seminal work into the forward/reserve allocation problem is from Hackman, Rosenblatt, and Olin (1990) who was the first to formulate the problem and solve it with a heuristic while Hackman and Platzman (1990) developed a better, near-optimal solution methodology of the same formulation. Gu, Goetschalckx, and McGinnis (2010) expand Hackman's and Platzman's solution technique by developing a branch-and-bound procedure which guarantees an optimal solution. Other research into the forward/reserve allocation problem typically focuses on special conditions to the initial formulation. Some examples include only allowing unit-load replenishments of the forward area (van den Berg et al. 1998), optimally determining the size of each zone in the warehouse simultaneously as the forward/reserve allocation problem (Heragu et al. 2005), and designing the forward/reserve area according to Lean principles (Kong and Masel 2008).

2.2.3 Knapsack problem literature

The final set of literature which was reviewed for its application to the mobile food retailer product mix problem is knapsack problem (KP) optimization models. Since KPs are one of the most well-studied NP-Hard problems, the review in this dissertation cannot cover the full breadth of the existing research. Instead, interested readers are recommended to refer to existing literature reviews such as one of the first KP reviews by Salkin and De Kluyver (1975), a review of exact solution methodologies by Dudziński and Walukiewicz (1987) and Martello, Pisinger, and Toth (2000), and a review of heuristics of KP variants by Wilbaut, Hanafi, and Salhi (2008).

Given that existing KP literature covers multiple variants, algorithms, and modeling methodologies, the mobile food retailer product mix problem was preliminarily modeled to determine the type of KP models which are most applicable to the current problem. From this preliminary modeling, it was determined that any mathematical model for a mobile food retailer would at least have to incorporate one demand constraint in the formulation. Within KP literature, a demand constraint is an additional requirement imposed on the model which requires that a weighted summation of the decision variables meets or exceeds a given, independent positive threshold (Wilbaut, Hanafi, and Salhi 2008). This differs from the typical knapsack constraint which requires a different weighted summation of the decision variables to meet or not exceed a given, independent positive threshold. KP problem which incorporate a knapsack constraint and a demand constraint are commonly referred to as demand-constrained KPs (DKPs). A DKP is needed for the mobile food retailer product mix problem as these retailers will require their stocked product mix to exceed a given nutritional threshold or to exceed a given profit/revenue threshold. It is not always necessary that both of these be included as constraints, as they can be modeled as objectives in some circumstances, but a majority of retailers will have one or both requirements. Hence, existing DKP and similar literature will be reviewed followed by solution algorithms for the DKP and related KPs whose solution algorithms may be applicable to the DKP.

To date, there are is no technical literature on specialized solution methods for solving the DKP. Instead, most research focuses on the DKP variant called the multi-demand, multidimensional knapsack problem (MDMKP) where multiple knapsack constraints and multiple demand constraints can be included in one model. Examples of applied MDMKPs include the project selection problem from Beaujon, Marin, and McDonald (2001), the obnoxious facility location problem from Cappanera, Gallo, and Maffioli (2004), and the sea cargo mix problem from Ang, Cao, and Ye (2007). Due to the size of these problems, no exact solution methods exist and any heuristic solution algorithms typically only focus on small to average sized problems. Such work includes the Tabu Search procedures from Cappanera and Trubian (2005) and Arntzen, Hvattum, and Løkketangen (2006), the Scatter Search procedure from Hvattum and Løkketangen (2007), the Alternating Control Tree procedure from Hvattum et al. (2010), and the dominance procedure from Balachandar and Kannan (2011). Of these algorithms, only the Alternating Control Tree procedure by Hvattum and Løkketangen can guarantee optimality but commercial solvers must be used to solve the procedure's subproblem.

Given this limited research, research on modern solution algorithms for the KP and similar variants was also reviewed to see if solution algorithms for these problems could be applied to solve the DKP or the MDMKP. This research identified that the most advanced modern KP and KP variant solution methods utilize the concept of a 'core' set of variables. This concept was first introduced by Balas and Zemel (1980) who identified that there are only a small subset of KP decision variables whose optimal solution values differ between the binary solution and the relaxed linear solution. This subset of variables are hereafter referred to as the core or core variables. Furthermore, Balaz and Zemel identified that when all of the variables are sorted according to their objective-to-constraint coefficient ratios (hereafter referred to as efficiency measures), these core items are likely to be listed closer to those item(s) whose linear solution value(s) is non-binary (hereafter referred to as the break item(s)). The break items are therefore included in the core set of variables as they are likely non-binary.

Numerous solution algorithms have utilized this property to solve the binary KP. Most notably, Pisinger (1995a) developed a depth-first, branch-and-bound procedure which prioritized branching at variables close to the break item as measured by the variable's efficiency measures. This methodology of prioritizing branching according to the break item is referred to as the Expanding Core technique. This algorithm was later updated to the breadth-first Expanding Core procedure which proved to be more efficient (Pisinger 1997). Given the addition of cardinality constraints by Martello and Toth (1997) which restricted the feasible region to ensure that at least and at most a certain number of variables are included, Martello, Pisinger, and Toth (1999) introduced the best binary KP solution algorithm to date which combines the cardinality constraints with the Expanding Core procedure. Other modern solution methods are summarized by Martello, Pisinger, and Toth (2000).

Other recent advances have focused on expanding the concept of core variables and efficiency measures to KP variants. The greatest contribution of such research is in the development of efficiency measures for problems with multiple knapsack constraints. These types of measures were first introduced by Dobson (1982) who used a measure which was the ratio of the objective coefficient over the sum of the constraint coefficients. These measures have since been updated to feature a weighted sum of the constraint coefficients, typically weighted by the optimal dual variables, as demonstrated by Puchinger, Raidl, and Pferschy (2010), Angelelli, Mansini, and Speranza (2010), and Della Croce and Grosso (2012). In addition, either the efficiency measure for MKPs or the measure for standard KPs have been used to solve other KP variants including an equality KP (Volgenant and Marsman 1998), bounded KP (Pisinger 2000), unbounded KP (Martello and Toth 1990), multiple-choice KP (Pisinger 1995b), multiple-choice MKP (Ghasemi and Razzazi 2011), and multi-objective KP (Gomes da Silva, Clímaco, and Rui Figueira 2008; Mavrotas, Rui Figueira, and Florios 2009; Lust and Teghem 2012). For those interested in more information, concise reviews exist for solving KPs or their variants using core approaches, either exactly (Dudziński and Walukiewicz 1987; Martello, Pisinger, and Toth 2000) or heuristically (Wilbaut, Hanafi, and Salhi 2008).

Based on the aforementioned literature, I selected to model the mobile food retailer product mix decision as a either a DKP or MDMKP. Modeling the mobile food retailer product mix problem similar to the models used in the supermarket product mix literature would be difficult given the nuances of the mobile food retailer with respect to customer costs and the grocery item health. Similarly, the numerous constraints used in the TFP for each type of nutrient are more than what is required for mobile food retailers and moderate to large sized problems modeled using this technique may be difficult to solve efficiently. Finally, the warehousing literature was discarded as a feasible approach for modeling the mobile retailer product mix problem because defining the cost of having to visit the supermarket, e.g. reserve area, as opposed to the mobile retailer, e.g. forward area, would be highly subjective and may vary based on the retailer location. These challenges make all of the aforementioned techniques poor modeling choices.

Instead, the MDMKP was initially selected as the ideal choice for modeling the mobile food retailer product mix problem due to its flexibility to model any KP variant so long as it had a single linear objective function and linear constraints of any quantity and type. This flexibility is especially desirable for the mobile food retailer product mix problem as not all retailers will have the same needs and requirements due to differences in their communities and vehicle. Hence, improved solution methods for the MDMKP were first developed as the existing algorithms do not incorporate many of the modern advances in KP solution methods such as the use of core variables and efficiency measures. These developed solution methodologies are detailed in Chapter 3.

Given these solutions, the DKP was also selected as a possible model for the mobile food retailer product mix problem. The DKP would be suitable for those retailers whose only limitations are the size of the vehicle and the need for a sufficient profit margin or the need for the stocked product mix to meet a given ‘healthiness’ threshold. In such a case, having a dedicated DKP solution method is advantageous as an algorithm developed specifically for the DKP will likely provide higher quality solutions in less time compared with applying a MDMKP solver to DKP instances. The first exact DKP solution method was therefore developed as part of this research effort and is detailed in Chapter 4.

2.3 Mobile food retailer scheduling and routing literature

To date, the mobile food retailer scheduling and routing problem is the sole aspect of this dissertation which has been addressed within technical literature. For example, Algert, Agrawal, and Lewis (2006) identified potential areas to service by clustering demand, but provided no discussion to the routing aspect of the problem. Additionally, only the residential location of the population was considered and which subset of clustered locations should be served was never addressed. The other primary research on this topic is from Widener, Metcalf, and Bar-Yam (2012; 2013) who identified which customers lack access to supermarkets and which of these groups should be served in an optimal solution. Again, no discussion is given to routing the vehicle and only a subset of the visitable locations are considered in their formulation. Hence, more research is needed to assist a mobile food retailer coordinator in making the optimal routing decision.

Prior to discussing additional literature relevant to the mobile food retailer scheduling and routing problem, it is important to summarize the key requirements and considerations when constructing the ideal routing plan. One of the most important requirements is that feasible routes have to be constructed. This implies the vehicles must stop and start at a centralized depot or warehouse, a vehicle can only service locations so long as it has sufficient inventory, and the vehicle must return to the depot by the close of business. In addition, the routes should be designed such that the maximum revenue is captured. For these applications it is assumed that serving the maximum revenue is equivalent to serving the demand of as many food desert residents as possible.

In addition to these requirements, there are additional considerations which may be valid based on the community served by the retailer. For instance, it is not valid to assume that the retailer is capable of serving the entire candidate set of locations as the cumulative demand of all of these locations may exceed the capacity of the retailer(s). Instead, the developed solution methodology must be able to only serve the ideal subset of these customers if required. These locations may also have strict time windows for service which the retailer must satisfy if it is to service demand. Finally, it may be possible for customers to travel between locations if its direct location is not serviced.

Based on requirements, the mobile food retailer scheduling and routing problem is modeled as the Covering Capacitated Vehicle Routing Problem (CCVRP) which is variant of the traditional Capacitated Vehicle Routing Problem (CVRP). The CCVRP includes all of the traditional model elements of the CVRP with the sole exception that it is assumed that the vehicle in the CCVRP can satisfy demand at a service location by stopping at a different service location so long as the two locations are within an established distance threshold. The motivation for using the CCVRP to model the mobile food retailer scheduling and routing problem is that the covering mechanic of the CCVRP is equivalent to customers traveling between service locations to travel to a nearby mobile food retailer assuming that such a retailer does not directly service that customer's initial location. It is even possible to extend this variant by adding assumptions which state that not all locations must be serviced (in the case where more demand exists than can be satisfied by all of the developed routes) and that all locations have restrictive time windows.

Given this modeling approach, the literature on CVRP variants similar to the CCVRP employed in this research will be discussed next. Following this will be a summary of solution algorithms for these variants as well as solution algorithms for general CVRPs which are relevant to the solution methods for the CCVRPs discussed in Chapters 5 and 6.

One set of literature which has problems that resemble the CCVRP is routing literature on public service vehicles such as city buses, city trains, and school buses. With respect to the two former areas of literature, Schöbel (2012) provides a recent overview of relevant literature for the planning of all public transportation systems which is recommended for readers who are interested in a thorough discussion of the topic. Literature into these topics can be categorized into two categories: cost-oriented models and passenger-oriented models. With respect to the routing problem faced by mobile food retailers, literature on cost-oriented models is the most applicable as the passenger-oriented approaches typically introduce additional considerations such as minimizing transfers, minimizing traveling time (time which includes penalties for transfers), or other considerations which are not necessary for mobile healthy food retailers. The discussion that follows will therefore omit passenger-oriented modeling approaches.

With respect to the planning of railway public transportation systems, one the earlier modern solution methods for solving the problem is from Bussieck, Kreuzer, and Zimmermann (1996) who utilized relaxations and cutting planes to determine heuristic solutions for the ideal rail network. Additional literature on the routing of public rail lines includes Claessens, van Dijk, and Zwaneveld (1998) who incorporated train length into their decision making process and solved the problem to optimal using a Branch-and-Bound procedure, Goossens, van Hoesel, and Kroon (2004) who developed a Branch-and-Cut procedure, Bussieck, Lindner, and Lübbecke (2004) who created a fast heuristic algorithm based on relaxations and variable fixing, and Laporte et al. (2005) who developed several heuristics for the creation of a new rapid transit line. With respect to the planning of bus routes, the literature is again extensive. Most modern literature focuses on metaheuristics for the problem as demonstrated by Euchi and Mraihi (2012) who developed an ant colony algorithm, Pattnaik, Mohan, and Tom (1998) who developed a genetic algorithm, and Fan and Mumford (2010) who developed a simulated annealing algorithm. Other noteworthy modern research into this topic is from Soumis, Desrosiers, and Desrochers (1984) who developed an exact algorithm (which was only tested on one example) and Yan and Chen (2002) who utilized Lagrangian relaxation and flow decomposition algorithms.

While these cited examples are only a small subset of the literature on the planning of public transit routes, it was enough of a review to demonstrate that this literature is not related to the problems faced by mobile food retailers. For instance, a major aspect of the planning of urban public transit lines is the frequency the line is travelled. Since mobile food retailers do not have to place the same emphasis on how frequently any one route is traveled, it was determined that most of this literature is not applicable to the current research. However, it can be argued that the frequency that a route is serviced is potentially relevant to mobile food retailers as it may be possible for a mobile retailer to have a set plan of routes which are revisited every week, bi-weekly, or once a month. This level of tactical planning is currently out of scope for this research. In addition, many retailers do not keep adhere to one route consistently over time as new stops are frequently introduced and trialed. This approach would therefore not be relevant to such retailers which includes the retailer that serves as the basis of the case study in Chapter 7.

In comparison to this research, literature on urban school bus planning does not have the same ‘frequency’ considerations as the developed routes are only used once per day (or twice to drop students off). Hence, literature from this area, specifically for urban bus routes where it is assumed that children's houses will not be directly visited but nearby sites will be utilized, is summarized. A review of this literature is provided by Park and Kim (2010) who cite that most of the literature on urban school bus routing does not consider the assignment of children to bus stops and the routing of the buses in one model. Instead, most of this literature solves these problems in two separate phases, as demonstrated by one of the earliest works in this area by Bodin and Berman (1979), which leads to suboptimality. The subset of this literature which addresses both problems simultaneously is therefore summarized next.

One of the first examples of such research is from Bowerman, Hall, and Calamai (1995) who developed a multi-objective approach for this problem which seeks to minimize the number of routes and total route length while also balancing the loads and lengths of the routes. Bowerman, Hall, and Calamai adapt the commonly used Allocation-Routing-Location (ARL) heuristic, which clusters students and potential bus stops prior to developing routes through each cluster, by performing the clustering using an adapted VRP heuristic and then perform the routing through the cluster using an adaptation of the COVTOUR algorithm from Current and Schilling (1989). A similar approach is used earlier by Chapleau, Ferland, and Rousseau (1985) with the exception that during the routing phase, it is assumed students are already assigned to a specific bus stop (compared to the algorithm by Bowerman, Hall, and Calamai where students are not yet assigned to stops at the end of the clustering phase). The next major approach for this problem is from Schittekat, Sevaux, and Sörenson (2006) who introduced a method to generate test instances and solved their instances by using commercial integer programming solvers by relaxing the subtour constraints and adding them as necessary.

The issue with this approach is that only small problems were solvable and most were not solvable within a reasonable amount of time. A more technical approach was introduced by Riera-Ledesma and Salazar-González (2012) who utilized a Branch-and-Cut algorithm to solve instances with up to 100 stops and users but many took up to and over 2 hours to solve. Schittekat et al. (2013) later revisited the problem by developing a parameter-free metaheuristic which can solve problems with up to 800 students and 80 stops.

While many of these problems are similar to the problem faced by mobile food retailers, they are not directly applicable. With respect to the two exact algorithms, Schittekat, Sevaux, and Sörenson (2006) can only solve small problems and is rather inefficient due to its basic approach while the algorithm from Riera-Ledesma and Salazar-González (2012) is applicable but splits the visitable locations from the customer locations. With respect to the heuristic approaches, they feature components which are not applicable to mobile retailers such as multiple objectives based on the travel time of the students or load balancing between the buses (Bowerman, Hall, and Calamai 1995). Given these issues, problems which are similar to the CCVRP but do not focus on a specific application were investigated and are summarized next.

The CCVRP combines elements from two classical problem: the CVRP and the set covering problem assuming that all customers can be serviced. If it is impossible to service all customers than the CCVRP is a combination of the CVRP and the maximal covering problem. Similar combinations of routing and covering problem have been addressed by numerous prior researchers. These include the set covering shortest path problem by Current, ReVelle, and Cohon (1984), the bi-maximal coverage shortest path problem by Current, ReVelle, and Cohon (1985), the p-median shortest path problem by Current, ReVelle, and Cohon (1987), the maximal coverage Traveling Salesman Problem (TSP) by Current and Schilling (1989; 1994), and the p-median TSP by Current, Pirkul, and Rolland (1994). Other examples which are similar to the CCVRP include the application focused research from Boffey and Narula (1998) who developed a multi-path, maximal covering formulation to plan subway routes, research from Wu and Murray (2005) who developed a maximal coverage shortest path problem to reroute an existing transit system, and research from Mohaymany and Pirnazar (2007) who routed vehicle covering paths to assist in emergency evacuation routes after an earthquake.

None of the aforementioned research is directly applicable to the CCVRP as they all incorporate different routing and/or covering problems. Instead, the most similar research comes from two separate research studies. The first is from Akinc and Srikanth (1992) who developed a model which is identical to the CCVRP except that it required all customers to be serviced (therefore it is not applicable to mobile food retailers who cannot service all their demand) and it introduced a penalty for serving demand from a distance. The second research study is from Halper and Raghavan (2011) who developed a CCVRP in which service rewards were modeled as a continuous function. This is not applicable to the CCVRP as it can only capture demand once upon arrival/service to any location. Hence, new research is needed to develop models and solution algorithms for the CCVRP applied to mobile food retailers.

Given the development of the mathematical models for the CCVRP (provided in Chapter 5 and 6), new solution algorithms need to be developed as commercial solvers cannot easily solve routing problem. In general, there is no preferred techniques to solving the CCVRP and similar problems. For example, all of the aforementioned problems are solved exactly or heuristically via combinations of Lagrangian relaxation of the covering constraints, linear relaxation, branch-and-bound, graph transformation, local search procedures, and/or problem separation where the covering problem and routing problem were solved separately to obtain a heuristic approximation. Hence, general CVRP literature was reviewed for both exact and heuristic solution procedures which may provide insight into preferred solution strategies.

With respect to the exact solution methodologies, there are three primary techniques employed to solve the CVRP to optimality: branch-and-bound, branch-and-cut, and column generation/set covering algorithms. The latter two of these techniques are preferred in modern solution methods. An overview of these three techniques is provided in a historical context by Semet, Toth and Vigo (2014) while Poggi and Uchoa (2014) updates these methods with modern advances.

Of these methodologies, the column generation technique was implemented as it the most readily adaptable to the CCVRP. For this approach, the CCVRP (or CVRP in the case of the cited literature) is transformed into an equivalent set-covering problem such that each variable represents a feasible vehicle route and covering plan. In the case of the CVRP, each variable only represents a feasible vehicle route. The problem is initially solved over a small set of these routes, but more are generated as needed until an optimal solution is obtained. There are numerous methodologies to generate these routes including the technique from Agarwal, Mathur, and Salkin (1989) who generated routes through branch-and-bound, Bixby, Coullard, and Simchi-Levi (1997) who calculated a lower bound via a prize-collecting TSP and Desrochers, Desrosiers, and Solomon (1992) who developed a dynamic programming algorithm. Ultimately, the technique from Agarwal, Mathur, and Salkin (1989) was implemented to solve the CCVRP exactly and the exact procedure and results are shown in Chapter 5.

With respect to solving the CCVRP via heuristics, there are three main methodologies: clustering-based heuristics, improvement-based heuristics, and metaheuristics. For the clustering-based heuristics, a two phase approach is typically employed. Commonly, the first phase clusters customers together and the second phase develops routes through these sets of clustered customers. One such technique is from Fisher and Jaikumar (1981) who first clustered locations using a generalized assignment problem prior to solving a TSP through the clustered locations. Similar approaches include the Sweep heuristic (Gillett and Miller 1974) which is described in detail in Chapter 6 as well as the Petal heuristic (Renaud, Boctor, and Laporte 1996) and Taillard’s heuristic (Taillard 1993) which are advancements on the basic Sweep algorithm as they permit more flexibility with respect to location partitioning. Note that it is also possible to first generate routes (i.e. a large tour through all customers) in the first phase and then cluster in the second phase by separating this tour into separate routes, but the approach is not commonly found in literature.

The improvement-based heuristics are focused on developing an initial set of routes, and then performing a set of operations which aim to improve the routes until some stopping condition is reached. The most popular algorithm is the Savings algorithm by Clarke and Wright (Clarke and Wright 1964) where the problem is initialized by creating a route for each location. These routes are then merged together in a greedy manner such that the routes which result in the greatest cost savings are merged together first. This process continues until all possible mergers have been evaluated. These routes are then enhanced using inter-route improvement exchanges which transfer a location or a set of locations between created routes. It is also common to complete intra-route improvement operations which are common to TSP heuristics such as the 2-opt algorithm which tests all possible swaps of two edges in a TSP route. If the swap would result in a shorter route, it is completed and the process is restarted. These improvements terminate once all possible swaps have been investigated and no shorter routes are possible.

The final set of common heuristics for the CVRP are metaheuristics such as the Ant Colony System (ACS), Tabu Search, or Genetic Algorithm procedure. For this implementation of the CCVRP, it was decided that the ACS procedure is the most relevant to the CCVRP. The full motivation for this decision is provided in Chapter 6. The remaining discussion in this section only serves to summarize the ACS approach and the relevant literature.

The ACS procedure is motivated by the foraging behavior of ants in a colony. When an ant leaves the colony in search of food, they do so by following the pheromone trails of ants which foraged previously. The more frequently this trail is followed, the more likely an ant is to travel along this path as the pheromones become stronger from all of the prior ants. The ACS procedure simulates this behavior by assigning a pheromone level to every edge in the network. Then a set of ants are allowed to visit all locations in the network such that edges with a greater pheromone level are more likely to be traversed by the ants. These paths are then split to create vehicle routes. The edges along the path followed by the best ant(s) have their pheromone levels increased while all others have their levels evaporated (i.e. returned to their starting values). This process is repeated until a stopping criterion is met and the best set of vehicle routes found by an ant is returned as the solution.

This general procedure of routing the ants, creating vehicle routes, and updating pheromones is common to all CVRP ACS applications. However, there is a variety of methodologies which are employed for each of these steps. For instance, the early Ant Colony Optimization (ACO) procedure by Bullnheimer, Hartl, and Strauss (1999) only used a probabilistic function when determining the next location for the ant while Bell and McMullen (2004) use a two level approach which probabilistically decides between the probabilistic approach and selecting the path with maximum pheromone value (which defines the difference between the ACS approach and the ACO approach). Furthermore, recent ACS procedures have incorporated elements from other heuristics such as a genetic modification procedure as demonstrated by Bin, Zhong-Zhen, and Baozhen (2009) or scatter search procedures as demonstrated by Zhang and Tang (2009). The ACS has also been applied to CVRP variants such as the VRP with time windows (Ding et al. 2012), the time dependent VRP (Donati et al. 2008), the VRP with multiple time windows (Favaretto, Moretti, and Pellegrini 2007), and the VRP with pickup/delivery (Gajpal and Abad 2009), but never to a CVRP with the capability to cover customers from nearby locations.

Given the quantity of heuristic solution methods, multiple heuristic solution methods were developed to solve the CCVRP: a Greedy procedure, Savings procedure, Sweep procedure, and ACS procedure. These methods are discussed in Chapter 6. The advantage of these procedures, in comparison to the exact procedure developed for Chapter 5, is that they are able to solve larger problems in a much shorter amount of time.

This disadvantage of these methodologies is that the solution cannot guaranteed to be optimal, but the results in Chapter 6 will demonstrate the identified routing solutions are still of high quality.

2.4 Conclusion

The objective of this chapter was to demonstrate the need for addressing issues related to mobile food retailers and to provide justification on how to address two key operational issues for mobile food retailers. With respect to the former, there are clear issues with food desert research due to the noted inter-study differences. However, the quantity of studies which have identified statistical disparities in food access among different demographic groups provide a strong argument that providing better food access for traditional food desert residents is a justifiable use of funds and resources. Furthermore, studies investigating the effects of poor food access (which suffer from the same inter-study differences as the studies investigating disparities in access) have all identified that having better access to healthy and affordable foods does not worsen health outcomes. In fact, numerous studies have identified that improving access can improve health outcomes for at-risk demographics. Hence, intervention techniques, such as mobile retailers, can have a significant and important effect on the communities they serve.

As stated in Chapter 1, two operational decisions will be the main focal points of this dissertation. The first operational decision is determining the optimal product mix to stock on a mobile retailer. After a review of applicable modeling techniques, it was determined that a knapsack optimization model would be the ideal modeling choice so long as demand constraints were included. Demand constraints (which require a weight summation of the variables to exceed a given threshold) are necessary as many retailers are expected to require their product mix to meet or exceed given thresholds on nutrition or profits. To date, solution methods for demand constrained knapsack problems, both DKPs and MDMKPs, are limited and none of the existing procedures feature modern KP solution methodologies such as core variables or efficiency measures. The theoretical contributions of this dissertation are to expand these concepts to demand constrained knapsack problems. Specifically, three heuristic procedures for MDMKPs are discussed in Chapter 3 and an exact solution procedure for DKPs is discussed in Chapter 4.

For all of the solution procedures provided in Chapters 3 and 4, specific emphasis is placed on solution algorithms which do not require commercial software. The rationale for this goal is that these procedures can be provided to mobile food retailers to assist with the planning of their product mix with minimal need for external aid or software. Through such tools, mobile retailers will be able to balance their competing objectives of health, consumer cost, and retailer profit to determine a plan which is best meets their current operating conditions. To demonstrate how a mobile retailer could use such tools, the case study in Chapter 7 shows example analyses using both DKP and MDMKP formulations based on real operational data provided by a mobile food retailer. Such work has never before been completed and it demonstrates how mobile food retailers can increase their profits thereby better ensuring their economic sustainability.

The second operational decision which serves as the focal point of this dissertation is the mobile food retailer routing and scheduling problem. As discussed, this is the sole area of the mobile food retailer problem which has been addressed by technical literature but prior research solely focused on clustering methodologies to determine possible service locations and none address the routing of the mobile retailer. Based on existing technical literature, it was determined that a CCVRP model best represents the decisions faced by mobile food retailers. To date, research on this VRP variant has been limited as only one research effort has been identified which solves the problem exactly through Lagrangian relaxation (Akinc and Srikanth 1992) and no algorithms were identified which are dedicated heuristic procedures for the CCVRP. The theoretical contributions of this dissertation are to address these shortcomings by expanding the existing literature on the CCVRP. Specifically, a new exact algorithm based on column generation is discussed in Chapter 5 and four heuristic solution procedures for the CCVRP are discussed in Chapter 6.

Similarly to the research into the product mix decision, the emphasis of the research into the scheduling and routing problem is on the development of procedures which are readily available for any practitioner. Specifically, the techniques in Chapter 6 are designed such that no commercial software is needed while still providing efficient routing plans. Furthermore, the algorithms presented in this dissertation will be the first procedures for the design of mobile food retailer routes which simultaneously determine the clustering of service points and the routing of a vehicle through those service points. To demonstrate how a mobile retailer could use such tools, the case study in Chapter 7 develops routes through underserved south and west Phoenix communities using operational and collected data. The results from these tests demonstrate how mobile retailers can improve their economic sustainability through better route planning.

Parts of this literature review chapter were included in Wishon and Villalobos (2016a; 2016b).

CHAPTER 3

THE GENERIC MOBILE RETAILER PRODUCT MIX PROBLEM

Within this chapter, the solution algorithms for the generic mobile retailer product mix problem will be discussed. For this version of the problem, there are no assumptions on the number of requirements or restrictions on the product mix of a given retailer. Instead, the mobile food retailer is permitted to have as many constraints of any type so long as the retailer is deciding whether or not to stock a specific food item (but the quantity to stock is predetermined) and the objective is able to be formulated as a linear equation. Such an approach is needed as different retailers will have different requirements and restrictions on their product mix. For instance, one retailer may be concerned with ensuring their product mix is not too expensive for their customers while another may want to limit the number of substitutable goods which are simultaneously stocked on the retailer. Hence, solution algorithms applicable to all types of mobile retailer product mix problems is preferred.

As discussed in Chapter 2, the generic mobile retailer product mix problem can be modeled as a MDMKP. However, MDMKP literature is limited as it doesn't incorporate many of the most recent advances in solving KPs such as the concept of efficiency measures and core variables. Within this chapter, these concepts will be expanded to MDMKPs. Specifically, robust efficiency measures are presented which are now applicable to MDMKPs and bounded MDMKPs. A bounded MDMKP is a variant of the MDMKP where each variable is no longer binary but is instead a nonnegative integer which is less than a given upper bound. Based on these new measures, three new heuristic solution procedures will be presented: Fixed-Core algorithm, Kernel Search algorithm, and a Genetic algorithm. Computational tests are performed using these solution algorithms and the applicability of the techniques to the retailer product mix problem are discussed based on the results.

3.1 Efficiency measures for MDMKPs

The first step to developing new MDMKP solution algorithms was to expand the concept of efficiency measures to MDMKP. Currently, efficiency measures have only been developed for KP variants which do not include any demand constraints. To provide the MDMKP efficiency measures, consider the formulation for the general bounded MDMKP given below.

(MDMKP) Maximize: $z = \sum_{j=1}^{n} c_j x_j$ (3-1)

$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad \forall i \in \{1, \dots, m\}$ (3-2)

$\sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad \forall i \in \{m+1, \dots, m+q\}$ (3-3)

$x_j \in \{0, \dots, d_j\} \quad \forall j \{1, \dots, n\}$ (3-4)

In total, there are n decision variables denoted by x_j and the objective given by (3-1) is to maximize the summation of these variables weighted by c_j. The MDMKP is constrained by m knapsack constraints given as constraint set (3-2) and q demand constraints given as constraint set (3-3). Finally, each variable is bounded such that it can have any integer value between 0 and d_j as denoted by (3-4). The bounded form is used in this formulation since it is a more robust variant and the binary MDMKP is a special case where $d_j = 1$ for all j.

In this formulation, the only assumption is that $b_i \geq 0$ for all $i \in \{1, \dots, m+q\}$. Hence, c_j and a_{ij} for all i and j are unrestricted in sign which differs from prior knapsack problem research. If some b_i is negative, the constraint can be negated and substituted to the other constraint set. Furthermore, well-stated MDMKPs assume that $\sum_{j=1}^{n} d_j a_{ij} > b_i \; \forall i \in \{1, \dots, m+q\}$ since a violation for any $i \in \{1, \dots, m\}$ would imply that some knapsack constraint would never be violated and that a violation for any $i \in \{m+1, \dots, m+q\}$ would indicate that the problem is infeasible.

To develop the efficiency measures, the LP relaxation of the bounded MDMKP (LMDMKP) is required. The LMDMKP formulation is the same as the MDMKP except all integer variables are replaced with their linear counterparts, x_j^{LR}, and constraint set (3-4) is replaced with the set of constraints given in (3-5) and (3-6).

$$x_j^{LR} \leq d_j \qquad \forall j \in \{1, \dots, n\} \tag{3-5}$$

$$x_j^{LR} \in \mathbb{R}_+ \qquad \forall j \in \{1, \dots, n\} \tag{3-6}$$

Finally, the following formulation represents the dual formulation of the LMDMKP.

$$\text{(DLMDMKP) Minimize: } z_D^{LR} = \sum_{i=1}^{m} b_i u_i^{LR} - \sum_{i=m+1}^{m+q} b_i u_i^{LR} + \sum_{j=1}^{n} d_j v_j^{LR} \tag{3-7}$$

$$\sum_{i=1}^{m} a_{ij} u_i^{LR} - \sum_{i=m}^{m+q} a_{ij} u_i^{LR} + v_j^{LR} \geq c_j \qquad \forall j \in \{1, \dots, n\} \tag{3-8}$$

$$u_i^{LR} \in \mathbb{R}_+ \qquad \forall i \in \{1, \dots, m+q\} \tag{3-9}$$

$$v_j^{LR} \in \mathbb{R}_+ \qquad \forall j \in \{1, \dots, n\} \tag{3-10}$$

The DLMDMKP is defined by two sets of decision variables. The first set is denoted as u_i^{LR} for all $i \in \{1, \dots, m, m+1, \dots, m+q\}$ where the first m variables are associated with the knapsack constraints while the final q variables are associated with the demand constraints. The other set of dual variables is denoted as v_j^{LR} for all $i \in \{1, \dots, m\}$ which are associated with constraint set (3-5). All other coefficients are the same as the LMDMKP.

Given these formulations, the efficiency measures e_j for any decision variable j can be calculated as

$$e_j = \begin{cases} \sum_{i=1}^{m} a_{ij} u_i^{*LR} - \sum_{i=m+1}^{m+q} a_{ij} u_i^{*LR} / c_j & c_j > 0 \\ \sum_{i=1}^{m} a_{ij} u_i^{*LR} - \sum_{i=m+1}^{m+q} a_{ij} u_i^{*LR} + 1 & c_j = 0 \\ \sum_{i=m+1}^{m+q} a_{ij} u_i^{*LR} - \sum_{i=1}^{m} a_{ij} u_i^{*LR} / c_j + 2 & c_j < 0 \end{cases} \qquad (3\text{-}11)$$

where u_i^{*LR} is the optimal solution value for the dual variable u_i^{LR}.

Prior to demonstrating that these measures rank the decision variables according to their likelihood of being a core variable, other properties will first be demonstrated. First and foremost, the new robust efficiency measures provide an equivalent ranking of the decision variables as compared to existing measures for both standard KPs and MKPs. To demonstrate this equivalence, note that for either of these problems it is commonly assumed that all objective coefficients are positive in which only the first case in (3-11) must be considered. In addition, the second summation in the numerator of this calculation can be removed as there are no demand constraints in these formulations. This makes the new efficiency measures the inverse of the measures presented by Puchinger, Raidl, and Pferschy (2010) which therefore provide the same rankings but in the reversed order.

Furthermore, observe that the measures partition the variables according their optimal solutions values for the LMDMKP. This property is given in Theorem 1.

Theorem 1. For any $j \in \{1, \dots, n\}$, let x_j^{*LR} represent the optimal LMDMKP solution value. The following facts hold:

(i) If $x_j^{*LR} = d_j$, then $e_j \leq 1$

(ii) If $x_j^{*LR} = 0$, then $e_j \geq 1$

(iii) If $0 < x_j^{*LR} < d_j$, then $e_j = 1$

Proof: To prove Theorem 1, the following properties from complementary slackness of the LMDMKP and its dual formulation are required:

$$\left(\sum_{i=1}^{m} a_{ij}u_i^{*LR} - \sum_{i=m}^{m+q} a_{ij}u_i^{*LR} + v_j^{*LR} - c_j\right)x_j^{*LR} = 0, \tag{3-12}$$

$$\left(x_j^{*LR} - d_j\right)v_j^{*LR} = 0. \tag{3-13}$$

Consider any variable j such that $c_j < 0$. The other possible values for c_j will not be explicitly proven but can be easily demonstrated using similar reasoning. In the case of (i), $x_j^{*LR} = d_j$ and (3-12) implies that

$$\sum_{i=1}^{m} a_{ij}u_i^{*LR} - \sum_{i=m}^{m+q} a_{ij}u_i^{*LR} + v_j^{*LR} - c_j = 0. \tag{3-14}$$

Since $c_j < 0$, algebraic manipulation and $v_j^{*LR} \geq 0$ demonstrates that $e_j \leq 1$. In the case of (ii), $x_j^{*LR} = 0$ and (3-13) implies that $v_j^{*LR} = 0$. By substituting v_j^{*LR} in (3-8) along with $c_j < 0$, algebraic manipulation demonstrates that $e_j \geq 1$. Finally in the case of (iii), $x_j^{*LR} \neq d_j$ and (3-13) implies that $v_j^{*LR} = 0$. Since $x_j^{*LR} \neq 0$, then (3-14) must also hold in this case. After substituting v_j^{*LR} in (3-14), algebraic manipulation along with $c_j < 0$ demonstrates that $e_j = 1$. ■

Note that there is no biconditional equivalent to Theorem 1 due to the possible scenarios in which $e_j = 1$. However, it is possible to prove that if $e_j > 1$ then $x_j^{*LR} = d_j$ and that if $e_j < 1$ then $x_j^{*LR} = 0$ by using similar construction methods as Theorem 1 and statements (3-7) through (3-13).

The purpose of Theorem 1 is to demonstrate that (3-11) partitions the variables into three categories based on the solution to the LMDMKP. Observe that this is the same approach used by Dantzig (1957) for solving the linear KP. The most critical of these categories are those variables for which $e_j = 1$ as these variables are nearly guaranteed to be in the set of core variables as they are typically non-integer for the solution of LMDMKP. In KP terminology, these values are called the break items and serve as the starting point for many core-based solution methodologies.

Besides these break items, the other two sets from Theorem 1 partition the variables of a MDMKP based on their solution values. Within these partitions, the variables are ordered such that the further the variable's efficiency measure is from one, the more likely that variable will *not* be in the set of core variables. This can be practically demonstrated by varying the parameters in each efficiency measure such that the measure becomes smaller or larger. In every possible case, the changes coincide with making that variable more desirable in the solution (larger objective coefficient, smaller knapsack weights, and larger demand weights) if the measure value decreases or it makes the variable less desirable in the solution if the measure value increases. The clear advantage of this behavior is that more effort should be spent investigating variables whose measure value are near one as opposed to those which are further away. This concept serves as the fundamental basis for the conceptual tests that follow.

These measures can also be used as a simple test when there are new variables to consider in the formulation. For instance, assume an MDMKP has been solved but a new item is potentially introduced. Without solving the problem again, the item's efficiency measure can be estimated using the optimal multipliers from the prior tests. Since introducing this item, assuming the MDMKP instance is at least of moderate size, will not drastically change these multipliers, this estimate will indicate whether the item will heavily be considered for inclusion or exclusion or whether it will be similar to the break items. In the case it is likely to be excluded, the MDMKP instance does not have to be solved again as the solution will likely not change. In the case it is likely to be included, it is clearly recommended to solve the MDMKP instance again to observe the new solution. In the case where the item is similar to the break items, it is recommended the practitioners decide whether or not the instance should be re-solved since the item's inclusion may not be optimal. Even in the situation where including the item is optimal, it will not provide a large improvement to the solution value. Hence, these measures can be used as a screening mechanism for possible future items.

3.2 Preliminary tests for MDMKP efficiency measures

To demonstrate that the newly developed robust efficiency measures perform at least as well as the existing efficiency measures for KPs and MKPs, a set of small MDMKPs were solved such that the core variables could be identified. The results from similar tests can be seen in the efficiency measure tests outlined by Puchinger, Raidl, and Pferschy (2010) for MKPs and by Pisinger (1997) for KPs.

To test these measures, 1,000 binary MDMKPs were solved to obtain both the linear and integer solution for each test instance. Each instance included 200 variables (n), 10 knapsack constraints (m), and 10 demand constraints (q). The objective coefficients (c_j) were randomly sampled from an integer uniform distribution with range $[-10, 10]$ for all j. These coefficients were chosen such that all three cases of (3-11) would be well represented in the results. The constraint coefficients (a_{ij}) were randomly sampled from $[1, \ldots, 10]$ for all i and j. The constraint thresholds were calculated as

$$b_i = \alpha * \sum_{j=1}^{n} a_{ij} \qquad (3\text{-}15)$$

with $\alpha = 0.50$ for all i. Finally, $d_j = 1$ as each instance is a binary MDMKP. This random generation procedure extends similar uncorrelated test instance generation methods from Chu and Beasley (1998) who developed random MKP test instances. All instances were solved to optimality in CPLEX version 12.6. Identifying the optimal solution was possible due to the problem's small size and limited coefficient ranges.

The principal result from these tests is shown in Figure 2 which plots the frequency of observing a variable being in the core based on the variable's efficiency measure. The plot demonstrates that as a variable's efficiency measure deviates from one, it is less likely to be included in the core of the problem. Hence, these measures are an effective tool for clustering likely core variables around the set of break items. The same pattern occurs in all prior efficiency measures techniques for KP (Pisinger 1997) and MKP variants demonstrating that the new measures provide the same utility as the measures for traditional KPs.

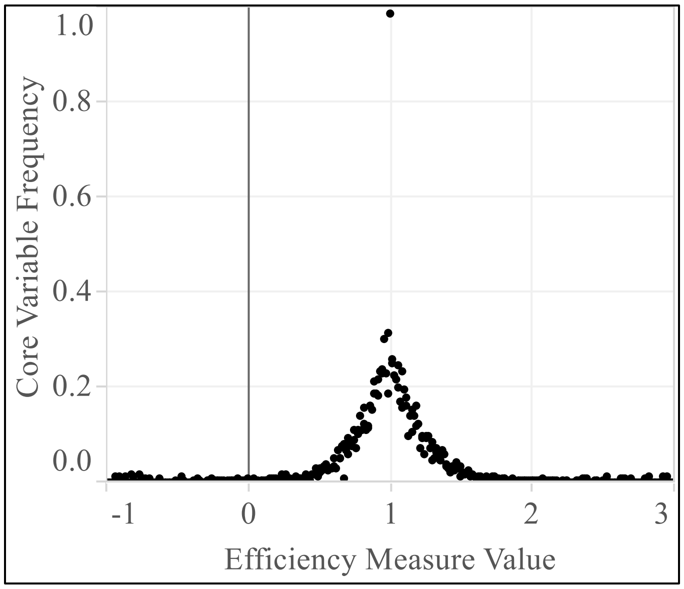


Figure 2. Core variable frequency by efficiency measure value

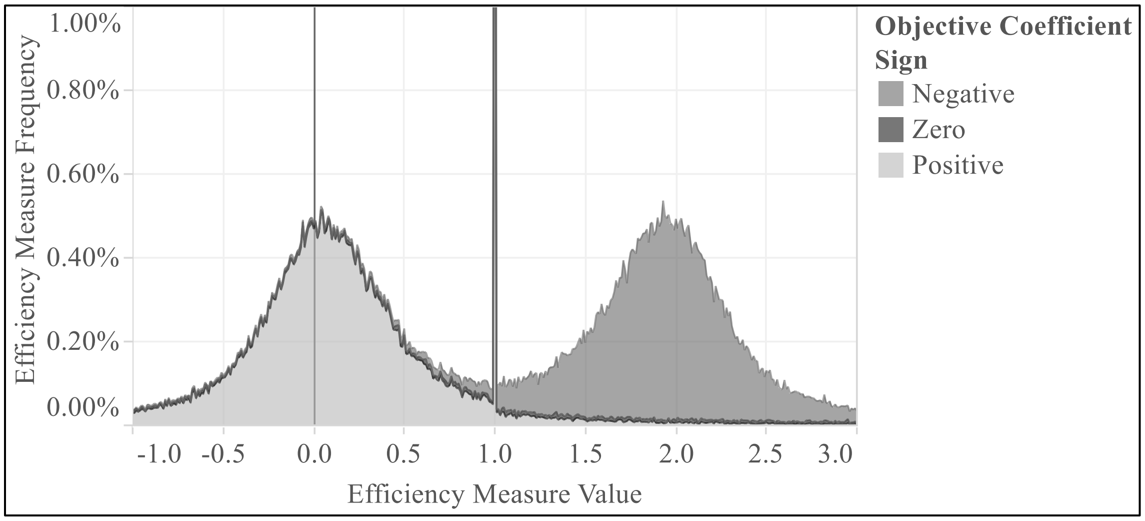


Figure 3. Efficiency measure frequency by objective coefficient sign

The advantage of the robust efficiency measures is further strengthened by Figure 3 which shows the frequency of observing a specific efficiency measure in the test problems based on the sign of c_j. In Figure 2, the efficiency measures of the core variables were clustered tightly around one and Figure 3 shows that the distribution of observed efficiency measures is bimodal with peaks at zero or two. Note there is a large spike at the efficiency measures associated with the break items (1.0), but this is to be expected as these are the only measures which are guaranteed to be observed in every problem. Hence, Figure 2 shows that nearly all of the observed core variables have efficiency measures between 0.5 and 1.5 while Figure 3 shows that observing such measures is not overly common with respect to all of the variables in the problem. This indicates that search techniques starting at the break items are likely to be efficient as there is not a significant quantity of variables with these measures. It should also be noted that the results shown in Figure 2 and Figure 3 are problem-dependent due to the nature of the random data generation and the simplicity of the problem. Hence, there is no guarantee that all MDMKPs will display this property but these tests indicate that such instances are possible.

3.3 MDMKP solution algorithms

To demonstrate the utility of the new efficiency measures, they have been applied to three solution algorithms for solving MDMKPs. Specifically, the measures will be employed in a Fixed-Core procedure, a Kernel Search procedure, and a Genetic Algorithm procedure. These solution methodologies were selected for two primary reasons. Principally, all three solution techniques were selected as they are heuristic procedures which have been applied to solve traditional KPs and MKPs, but never MDMKPs. Secondly, the efficiency measures will be employed in novel ways thereby expanding the state of the art for some of these heuristics. Specifically, the Kernel Search procedure, as far as I am aware, has never previously utilized efficiency measures for solving KP variants while a Genetic Algorithm has never had a mutation rate which is dependent on a variable's efficiency measure. Details of these modifications will be given in the appropriate sections.

Each of these techniques will be tested using a more robust data set than the data used in subsection 3.2. Specifically, randomly generated binary MDMKPs test instances were created with $n \in \{250,500,1000\}$ and $(m, q) \in \{(5,5\ ), (10,10\ ), (25, 25)\}$. All possible combinations of these values were used to generate test instances except for $n = 1000$ and $(m, q) = (25,25)$ which was excluded due to the complexity of the problem and the time required to solve the instances. For each instance, a_{ij} was randomly sampled from an integer uniform distribution over the range $[1,100]$ for all i and j. Likewise, c_j was randomly sampled from an integer uniform distribution over the range $\left[\sum_{i=1}^{m+q} a_{ij}/(m+q) - 50, \sum_{i=1}^{m+q} a_{ij}/(m+q) + 50\right]$ for all j. This data generation implies the objective coefficients will be slightly correlated with the constraint coefficients. Each b_i for $i \in \{1, \dots, m+q\}$ is calculated according to (3-15). For each $i \in \{m+1, .., m+q\}$, $\alpha = 0.50$ in (3-15) while $\alpha = 0.40$ for each $i \in \{1, .., m\}$ when $(m, q) = (5,5)$ and $\alpha = 0.47$ for each $i \in \{1, \dots, m\}$ for all other values of (m, q). These values for α differ based on the number of constraints due to feasibility challenges when m and q increase. In total, ten test instances were created for each of the parameter settings. This data generation methodology is similar to those employed by Puchinger, Raidl, and Pferschy (2010) but future research and tests are recommended to test application-based data as well as to test other data generation techniques such as negative values for a_{ij}, differing levels of coefficient correlation, and larger test instances with respect to both n and (m, q).

To establish a benchmark, each test instance will be solved using two existing methodologies. The first is CPLEX which was selected as it is assumed commercial software is available to most practitioners. Hence, the results from CPLEX are hereafter referred to as the 'base case' results. CPLEX was given the whole instance to solve and terminated once the difference between the best global integer solution and the linear relaxation of the best remaining node was less than 0.15% or until eight hours had elapsed.

The second benchmark methodology is the Alternating Control Tree (ACT) procedure from Hvattum et al. (2010). The ACT methodology was selected as it is currently the best approach for solving MDMKPs in the literature. For summary, the ACT procedure is an iterative process which continually solves the MDMKP linear relaxation and then solves a reduced binary subproblem based on this solution. This binary problem updates the current lower bound if possible and introduces cuts to the linear relaxation. This process is continued until the linear relaxation solution is less than the current lower bound. For this implementation, CPLEX was used as the technique to solve the binary subproblem. The algorithm was terminated after eight hours if the terminating condition was not met and all other parameters were set as recommended by Hvattum et al. It should be noted that the results from Hvattum et al. identified that the best subproblem solution method is to use a combination of CPLEX and their Scatter Search heuristic. CPLEX was selected in this research as it is easier to implement and provided results which are only slightly worse than the Scatter Search methodology as reported by Hvattum et al. I do not believe this shortcoming drastically alters the conclusions found in subsection 3.4 but future research may seek to compare the methods in this section with the improved subproblem solver. The full results of this methodology are shown in subsection 3.4 as the remainder of the current section will introduce the heuristics and compare their results solely to the base case.

Finally, the computational tests performed in the following subsections were not conducted with the aim to exhaustively study all tuning parameters in each of the presented heuristics. Specifically, no tuning parameters are studied for the Genetic Algorithm, the impact of varying the Fixed-Core size is tested in the Fixed-Core algorithm, and the number of buckets is tested in the Kernel Search algorithm. This is a clear shortcoming of the presented work, but the impact of varying tuning parameters is discussed when appropriate. Since varying most of these tuning parameters will have an obvious impact on the solution procedure (i.e. higher solution quality at the cost of longer solution times) and problems will have to be re-tuned if any of the data parameters are changed (n, m, q, coefficient correlations, α, etc.), I believe the impact of this shortcoming on the discussed conclusions is minimal since future researchers would have to perform computational tests to set tuning parameters based on their problem instances regardless of the values recommended in this research.

3.3.1 Fixed-Core MDMKP algorithm

The first application of the new measures is to the Fixed-Core solution methodology. The Fixed-Core solution method is motivated by the observation that knowing the true core variables of a knapsack prior to solving the binary/integer model is impossible, but sorting the variables according to their efficiency measures (as was done in Figure 2) groups the most likely core variables together. By assuming that the core variables are within a specifically selected subset of the sorted variables, the problem can be reduced by setting all variables outside of this subset to their linear solution values. If the true core is within this subset, then an optimal solution for the reduced problem is optimal for the full problem. Otherwise, the solution to the reduced problem is near-optimal for the full problem.

To outline the Fixed-Core algorithm, assume that there are b break items and the size of the desired fixed core is δ. After solving the linear relaxation of the MDMKP and calculating all of the efficiency measures for each variable, sort the variables according to these measures. The fixed core is then created from the b break items and the sets of $\lfloor(\delta - b)/2\rfloor$ items to the left and right of the break items. All other variables are fixed to their linear solution values and the reduced MDMKP is solved via CPLEX using the same stopping criteria as the base case. In this application, 7 core sizes were tested: $\delta_A = 0.1n$, $\delta_B = 0.15n$, $\delta_C = 0.2n$, $\delta_D = 0.1n + 0.1(m + q)$, $\delta_E = 0.2n + 0.1(m + q)$, $\delta_F = 0.1n + 0.2(m + q)$, and $\delta_G = 0.2n + 0.2(m + q)$. This procedure is similar to techniques applied to KPs and MKPs as demonstrated by Puchinger, Raidl, and Pferschy (2010).

The results with respect to solution quality from the Fixed-Core experiments are shown in Table 1 and Table 2. Specifically, both tables show two quality measures aggregated over the 10 test instances for each value of n, m, q, and δ. Table 1 displays the count of test cases in which the Fixed-Core approach found an equal or better solution than the base case. If any of the test instances were not solvable for the Fixed-Core test instance, the number of instances that were solved is given in parenthesis. An instance being unsolvable could either be a function of having no feasible region or could be a result of CPLEX not identifying any feasible solution within the eight hour limit. Table 2 shows the ratio of the Fixed-Core objective value over the base case objective value averaged over all of the *solvable* Fixed-Core and base case instances.

Table 1. Count of instances where the fixed-core procedure equals or outperforms the base case (Count of feasible fixed-core test instances if less than 10)

(n, m, q)	δ_A	δ_B	δ_C	δ_D	δ_E	δ_F	δ_G
(250, 5, 5)	2	5	6	2	6	3	6
(250, 10, 10)	7	7	9	6	8	8	8
(250, 25, 25)	1 (1)	3 (3)	2 (4)	2 (5)	2 (7)	3 (3)	2 (7)
(500, 5, 5)	8	6	9	10	9	8	6
(500, 10, 10)	6	9	5	7	6	8	4
(500, 25, 25)	6 (9)	7	8	6 (8)	6	6	7
(1000, 5, 5)	4	5	3	3	7	4	6
(1000, 10, 10)	4	3	6	5	4	7	4

Table 2. Average ratio of the fixed-core solution value over the base case solution value

(n, m, q)	δ_A	δ_B	δ_C	δ_D	δ_E	δ_F	δ_G
(250, 5, 5)	0.999	1.000	1.000	0.999	1.000	0.999	1.000
(250, 10, 10)	1.000	1.000	1.000	1.000	1.000	1.000	1.000
(250, 25, 25)	1.002	1.006	1.005	1.002	0.997	1.004	0.996
(500, 5, 5)	1.000	1.000	1.000	1.000	1.000	1.000	1.000
(500, 10, 10)	1.000	1.000	1.000	1.000	1.000	1.000	1.000
(500, 25, 25)	1.002	1.001	1.003	1.002	1.001	1.002	1.002
(1000, 5, 5)	1.000	1.000	1.000	1.000	1.000	1.000	1.000
(1000, 10, 10)	1.000	1.000	1.000	1.000	1.000	1.000	1.000

The results from the Fixed-Core experiments with respect to solution time are shown in Table 3. The measures are reported as the average time ratio to solve the Fixed-Core test instances over the time required to solve the base case instance. Hence, values less than 100% indicate that the time required to solve the Fixed-Core problem were less on average than the time required to solve the base case.

Table 3. Average ratio of the fixed-core solution time over the base case solution time

(n, m, q)	δ_A	δ_B	δ_C	δ_D	δ_E	δ_F	δ_G
(250, 5, 5)	0.116	0.233	0.631	0.153	0.719	0.150	0.660
(250, 10, 10)	0.120	0.334	0.354	0.117	0.411	0.172	0.399
(250, 25, 25)	1.000	1.000	1.000	1.000	1.000	1.000	1.000
(500, 5, 5)	0.424	0.657	0.719	0.398	0.654	0.369	0.859
(500, 10, 10)	0.622	0.564	0.877	0.633	0.778	0.782	0.822
(500, 25, 25)	1.000	1.000	1.000	1.000	1.000	1.000	1.000
(1000, 5, 5)	0.564	0.542	0.488	0.460	0.747	0.413	0.595
(1000, 10, 10)	0.659	0.851	0.713	0.669	0.732	0.713	0.828

The results from Table 1 and Table 2 demonstrate that the solutions obtained from the Fixed-Core experiments are competitive when compared with the base case. With respect to Table 1, a vast majority of the problem instances and core sizes found that the Fixed-Core algorithm identified an equal or better solution than the base case as shown by all of the entries which are 5 or greater. Also observed in Table 1 is that the quality of the identified solution increased as the core size grew which is expected as larger core sizes place less restrictive assumptions on the feasible region of the full problem.

With respect to specific test instances, when $n = 250$ and there are 10 total constraints, the procedure did not perform well with small core sizes according to Table 1. This is likely because these core sizes contained the smallest count of variables across all tested instances and therefore had the least flexibility if all of the true core variables were not within the fixed core. The only other instances which performed poorly in Table 1 are those which had 50 combined constraints. This is clearly because the fixed core was too restrictive in some of these instances and made the problem infeasible. However, the results from Table 2 demonstrate that when feasible solutions were identified in these cases, the Fixed-Core methodology greatly outperformed the base case solution. Hence, so long as the highly constrained problem is not rendered infeasible by a smaller core size, reducing the problem is advantageous since it focuses the solution effort. It should also be noted that the base case methodology had feasibility issues on some of the more constrained instances. For instance, when $n = 250$ and there are 50 total constraints, the base case only identified feasible solutions in 4 out of the 10 instances. Hence, the largest core sizes for this test combination were able find feasible solutions in instances which were deemed infeasible in the base case.

The only other cases which provided poor results in Table 2 are when $n = 250$ and there were only 10 total constraints. This is likely because the problem was relatively easier to solve in the base case (i.e. the time limit stopping criteria was not reached) and a small fixed core has a higher likelihood of missing key core variables. Hence, if the problem is smaller, a larger fixed core may be advisable to avoid such issues.

While the results from Table 1 and Table 2 demonstrate that there are not drastic quality differences between the base case and the Fixed-Core algorithm outside of the feasibility challenges from the $(250,5,5)$ instances, Table 3 demonstrates that the Fixed-Core methodology reports significant time savings in comparison to the base case. Specifically, other than for the most highly constrained instances, the average Fixed-Core time savings is 55% over the base case. The only instances where time savings were not observed are during the most constrained problems as all of the instances with 50 combined constraints terminated at eight hours for both the base case and for the Fixed-Core algorithm. Even though this does not represent a time savings, the results from Table 2 show that the Fixed-Core methodology is more efficient at finding high quality solutions in these situations.

Overall, the Fixed-Core algorithm can find equivalent or better solutions compared with solving the full problem using commercial software and frequently in a shorter amount of time as it can more efficiently focus the solution procedure. With respect to problems with a small amount of constraints and variables, it is recommended that a larger fixed core is employed as these small problems may be too constrained by a smaller core. For all larger problems which are not highly constrained, the smallest fixed core size is highly competitive with respect to solution quality. Finally, highly constrained problem should first be approached with small fixed core sizes and if preprocessing identifies the problem is infeasible, the fixed core size should be increased until the problem can be solved.

3.3.2 MDMKP Kernel Search algorithm

The second application of the robust efficiency measures is to expand the work of Angelelli, Mansini, and Speranza (2010) to be applicable to solving MDMKPs. In their work, Angelelli, Mansini, and Speranza developed a procedure, called the Kernel Search, which is used to solve MKPs. The solution procedure starts by identifying a set of promising decision variables, analogous to the set of core variables, which are referred to as the kernel. This kernel is then expanded based on the results from small, integer programming subproblems. Once all subproblems have been completed, a final heuristic solution is obtained.

The adaption of Angelelli, Mansini, and Speranza's algorithm to solve MDMKPs is straightforward as the initial algorithm only needs minor changes. The adapted Kernel Search procedure for MDMKPs is outlined below and the same notation will be used to maintain consistency between the approaches. The differences between the Kernel Search for the MDMKP and the algorithm presented by Angelelli, Mansini, and Speranza are discussed later. Those interested in a more in-depth discussion of the original approach are referred to the original publication by Angelelli, Mansini and Speranza (2010).

To define the Kernel Search algorithm, let N represent the set of decision variables and let T_{max} represent the user-defined maximum computational time. Let $\boldsymbol{x}^*$ and z^* store the best identified solution and solution value respectively. Let $\{B_i\}$ represent a set of pairwise independent buckets containing all variables excluding the break items. The construction methodology for the buckets is explained shortly. Finally, let Λ (hereafter referred to as the kernel) represent a subset of N and let $MDMKP(\Lambda)$ represent solving the MDMKP instance assuming that all $N \notin \Lambda$ are fixed to their linear relaxation solution values and all variables in Λ are constrained to be binary.

Using these definitions, the Kernel Search algorithm for solving the MDMKP is as follows:

MDMKP Kernel Search

```
Solve LMDMKP
Sort $N$ according to $e_j$
Construct the following:
    *If* $e_j = 1$, *then* $x_j \in \Lambda$
    Split $N \setminus \Lambda$ into a sequence of pairwise independent buckets $\{B_i\}$
    Let $t = T_{max}/(|\{B_i\}| + 1)$
Solve $MDMKP(\Lambda)$ with time limit $t$ and update $\boldsymbol{x}^*$ and $z^*$ if feasible
*For all* $B_i$ *in* $\{B_i\}$
    Let $\Lambda_i = \Lambda \cup B_i$
    Solve $MDMKP(\Lambda_i)$ with time limit $t$ and added constraints:
        *If* $e_j > 1$ for all $j \in B_i$, *then* $\sum_{j \in B_i} x_j \geq 1$
        *If* $e_j < 1$ for all $j \in B_i$, *then* $\sum_{j \in B_i} x_j \leq |B_i| - 1$
        $z \geq z^*$
    *If* a feasible solution to $MDMKP(\Lambda_i)$ has been identified, *then*
        Update $\boldsymbol{x}^*$ and $z^*$ according to $MDMKP(\Lambda_i)$
        Let $\overline{\Lambda}_i \subseteq B_i$ represent any items whose solution differs between
            $MDMKP(\Lambda_i)$ and LDMKP
        $\Lambda = \Lambda \cup \overline{\Lambda}_i$
    *End if*
*End for*
```

This procedure has two key differences from the algorithm employed by Angelelli, Mansini and Speranza (2010). Most importantly, the Kernel Search for the MDMKP sorts the variables according to their efficiency measures while the algorithm used by Angelelli, Mansini and Speranza uses the reduced cost of the variable. The other difference is that Λ was initialized by Angelelli, Mansini and Speranza such that it contained all the items which were included in the solution to the linear relaxation. Testing the impact of these differences is outside of the scope of this work and future research could be conducted to understand the repercussions of these changes.

To construct $\{B_i\}$, let NB be a user-defined even number that represents the total number of buckets. Since all $j \in N$ are already sorted according to e_j at this phase of the Kernel Search procedure, assign all $j \in N$ such that $e_j < 1$ into $NB/2$ buckets where each bucket contains pairwise independent groupings of variables which are adjacent in the ordering. The same should be performed for all $j \in N$ such that $e_j > 1$ for the other $NB/2$ buckets. Within the Kernel Search procedure, these buckets can be investigated in any order, but this implementation analyzed buckets by selecting the next bucket such that its elements had efficiency measures closest to one compared with the remaining buckets which have yet to be analyzed.

In this research, two methodologies were tested for the size of each $B_i \in \{B_i\}$. The first methodology is that all buckets had a uniform size which is similar to the initial algorithm by Angelelli, Mansini and Speranza. The other is that the buckets whose variables had measures closest to one were smaller than those buckets whose variables had measures which were further from one. Specifically, an exponential approach was taken such that the buckets whose variables had measures which were one step further from one were twice the size as compared with the buckets whose variables had measures which were one step closer to one. For example, if there were only seven items whose measures exceeded one and six buckets were desired in total (three buckets would be allocated to contain these items), then the first tested bucket would contain the one item whose measure is closest to one, the next bucket would contain the two items whose measures are next closest to one, and the final bucket would contain the last four items whose measures are the furthest from one. This strategy was hypothesized to be better than the uniform approach as buckets which had variables whose efficiency measures were further from one were less likely to contain core variables. Hence, making these buckets larger could improve the efficiency of the Kernel Search approach as more of these unlikely core variables would be investigated at one time.

The Kernel Search was implemented in MATLAB 2013, all subproblems were solved via CPLEX, and all test instances described at the start of subsection 3.3 were tested. Each instance was solved for $NB \in \{12, 14, \dots, 28, 30\}$ in order to draw conclusions on ideal bucket sizes and T_{max} was set to 2,880 seconds for all test instances. This setting was used as the total computational time to solve all ten bucket sizes for one test instance would be eight hours. Therefore, if there is no clear dominance with respect to bucket size and all buckets must be investigated for the best solution, the computational burden of the Kernel Search procedure is equivalent to the time limit for the Fixed-Core and base case experiments.

The results from these tests instances are summarized in Table 4. Specifically, the results are reported for all tested combinations of n, m, and q and bucket construction methodologies (uniform vs. exponential). For each test instance, the objective value as a percentage of the base case objective value was calculated and then averaged over all values of NB. The grand average of these values for all ten test instances at each combination of (n, m, q) is listed in Table 4. In addition, the maximum ratio over all values of NB was identified for each test instance and the grand average of these ratios over all test ten instances at each combination of (n, m, q) is presented in the parentheses. The average best NB to use over all ten instances is also reported. Finally, the time to solve the exponential bucket implementation divided by the time to solve the uniform bucket implementation averaged over all instances and values for NB is provided in the last column of Table 4.

Table 4. Kernel search solution objective and time results

(n, m, q)	Uniform Buckets Ave Obj Ratio (Max Obj Ratio)	 Best *NB*	Exponential Buckets Ave Obj Ratio (Max Obj Ratio)	 Best *NB*	Exp. over Unif. Time Ratio
(250, 5, 5)	0.999 (1.000)	18	0.999 (0.999)	20	0.651
(250, 10, 10)	1.000 (1.000)	20	0.999 (1.000)	20	0.495
(250, 25, 25)	0.962 (0.986)	24	0.970 (0.987)	24	0.766
(500, 5, 5)	1.000 (1.000)	22	0.999 (1.000)	20	0.331
(500, 10, 10)	1.000 (1.000)	22	1.000 (1.000)	20	0.545
(500, 25, 25)	0.991 (1.000)	20	0.991 (1.000)	22	0.822
(1000, 5, 5)	1.000 (1.001)	22	1.000 (1.001)	18	0.650
(1000, 10, 10)	1.000 (1.000)	26	1.000 (1.000)	22	0.757

The results in Table 4 demonstrate that the Kernel Search method applied to MDMKPs is an effective solution technique. Specifically, all instances except those with fifty combined constraints were solvable with a high level of accuracy on average. Furthermore, if only the best bucket sizes are considered as shown in the parenthesis in Table 4, the average solution quality is equal to or better than the base case in a majority of the test instances regardless of the bucket strategy. In addition, eight out of the ten (250, 25, 25) instances were solved for at least one value of NB. This is better than the base case where only four of these instances were solved and better than the Fixed-Core approach where the largest core sizes only solved seven of these instances. Hence, the Kernel Search methodology appears to be a better approach for finding feasible solutions.

With respect to the ideal value for NB, the results in Table 4 do not permit clear conclusions. In general, if the value for n is constant, the best value for NB increases as the number of constraints increases. This pattern is observed for nearly all values of n and bucket construction methodologies. Furthermore, as n increases, the best value for NB increases if the number of constraints and bucket construction methodologies are kept constant except for the sole outlier using exponential buckets and 10 total constraints. Outside of these trends, knowing the best bucket size prior to solving the problem would be challenging to identify without extensive computational testing on a variety of problem types and parameters. Since such testing is not the singular objective of this research, it is advised that the implemented approach (using multiple values for NB and retaining the best) be followed until such work can be performed. Therefore, the discussion of the results to follow focuses solely on the numbers in parentheses from Table 4.

With respect to solution quality, either bucket construction methodology is preferred as they each provide competitive results. However, it should be noted that if another digit were to be shown in Table 4, the results favor the uniform bucket strategy in a majority of the cases. With respect to computational time, the last column of Table 4 clearly shows that the exponential bucket strategy is preferred for all test instances. Hence, unless solution quality is the sole factor in selecting a solution method, the exponential bucket strategy is recommended as it is significantly faster than using uniform buckets and finds the same or nearly the same results. The uniform bucket approach is therefore only recommended in those situations when a 0.01% or less improvement in the final solution value is more important than a 50% average time improvement.

3.3.3 MDMKP Genetic algorithm

The final demonstration of the applicability of the new efficiency measures is within a Genetic Algorithm (GA). GAs are a common methodology for solving both KPs and MKPs, but they have never been applied to the MDMKP. GAs for solving MKPs have specifically been well studied in recent years with the most noteworthy advancement from Chu and Beasley (1998) who utilized efficiency measures within the repair phase to encourage better evolution towards optimality.

The GA which follows, hereafter referred to as the Efficiency-Weighted GA (EWGA), is developed as evidence that GAs can be applied to MDMKPs as well as proof of concept that efficiency measures can be applied to the mutation phase of GAs. To outline the EWGA, let K^l represent the population at the lth iteration of the EWGA. Assume that there are p individuals during each iteration. Let $k \in K^l$ represent an individual of this population which is defined by the binary array $\boldsymbol{x}^k$. This array represents a solution vector for the MDMKP which is not necessarily feasible.

The most unique component of the EWGA is that each gene $j \in N$ (i.e. each decision variable) has a specific mutation rate r_j which represents the probability that a gene will mutate whenever the mutation procedure is performed. Specifically, the mutation procedure generates a continuous random number between zero and one for each gene. If this random number is less than r_j, the binary value for that gene is flipped. This mutation operation occurs at two points within the EWGA.

The first occurrence is after the initial population is created. To demonstrate the mutation rates at this phase, assume there is an MDMKP instance with 100 variables which are sorted according to their efficiency measures. Furthermore, assume there is only one break item which is the 50th item in this sorting. Within the EWGA, an initial population of p individuals is created by rounding the optimal linear solution value for the MDMKP instance. Hence, all p individuals represent the same solution to the binary MDMKP prior to mutation. With respect to the aforementioned example, let $r_{50} = 0.50$. Any break item has this mutation rate which represents an equal likelihood of being a zero or one in the initial population. Next, let $r_{49} = r_{51} = 0.05 + \exp\big(-1 - 1 * (1/16)\big)$, $r_{48} = r_{52} = 0.05 + \exp\big(-1 - 2 * (1/16)\big)$, and so forth. In general, $r_j = 0.05 + \exp\big(-1 - \hat{\jmath} * (1/16)\big)$ where $\hat{\jmath}$ represents the sorted distance to the break item set for variable $j \in N$ and $r_j = 0.50$ if j is a break item. This methodology makes the mutation rate for a variable exponentially decrease as the variable becomes less likely to be in the set of core variables until the mutation rates asymptotically approach 0.05 for those items furthest from the break items. The specific approach was implemented as early computational tests found it outperformed other techniques and parameters.

Note that these rates only apply when mutating the starting population. The other mutation procedure occurs after a new offspring is created. In this case, the mutation rate is scaled such that one gene is flipped on average in each offspring. Specifically, let $R = \sum_{j=1}^{n} r_j$ and then the new mutation rate for any variable j is $r_j' = r_j / R$. By using the same mutation methodology as before (i.e. generating random variables for each variable), the new mutation rate will ensure that one gene is flipped on average for each offspring.

Outside of this mutation procedure, the EWGA is similar to existing Genetic Algorithms seen in literature. Specifically, each individual k is scored based on its fitness measure

$$f_k = \sum_{j=1}^{n} c_j x_j^k + M\left[\sum_{i=1}^{m} u_i^{*LR} \min\left(0, b_i - \sum_{j=1}^{n} a_{ij} x_j^k\right) + \sum_{i=m+1}^{m+q} u_i^{*LR} \min\left(0, \sum_{j=1}^{n} a_{ij} x_j^k - b_i\right)\right] \tag{3-16}$$

which is a summation of the objective function value for k penalized by the weighted violation of all knapsack and demand constraints respectively. These violations are individually weighted by the optimal dual solution values corresponding to each constraint and globally weighted by a large value M. This penalty is required as there is no simple repair procedure possible for individuals who are infeasible for MDMKP problems.

For the parent selection methodology, a tournament method is employed which randomly selects two pairs of individuals from the current population. The best individuals, as measured by their fitness, from each of these pairs is then selected as the parents. The offspring is created by randomly selecting a parent to pass along their information for each gene. Once this is completed, the offspring is mutated by utilizing r_j' as described previously. This process is completed to create p offspring. If an offspring is created which is feasible and better than the current best feasible solution identified thus far, z^* and $\boldsymbol{x}^*$ are updated. Then, the best p' offspring are selected and joined with the best $p - p'$ parents to create a new population and the process is repeated. This is continued until I populations have been created without an improvement in z^*. The full EWGA procedure is given next.

Efficiency-Weighted Genetic Algorithm

```
Solve LMDMKP
Let $l = 0$, $z^* = -\infty$, and $\boldsymbol{x}^* = \emptyset$
Initialize $K^0$ of $p$ individuals by rounding solution of linear MDMKP relaxation
Mutate $K^0$ using initial mutation rate $r_j$
For each $j \in N$, let $r_j' = r_j / \sum_{j=1}^{n} r_j$
*While* $l \leq I$
    *For* $h = 1$ *to* $p$
        Select two pairs of individual from $K^l$
            Set the best individuals from each pair as the parents
            Create the $h$th offspring by randomly selecting genes from each parent
        Mutate the $h$th offspring according to $r_j$
    *End for*
    Create $K^{l+1}$ by selecting the best $p - p'$ individual from $K^l$ and selecting the best
        $p'$ individuals from the offspring population
    Let $z' = \max_{k \in K^{l+1}} \left\{ \sum_{j=1}^{n} c_j x_j^k \,\middle|\, \boldsymbol{x}^k \text{ is a feasible solution} \right\}$
    *If* $z' > z^*$, *then* $l = 0$ and update $z^*$ and $\boldsymbol{x}^*$
*End while*
```

The EWGA was implemented in MATLAB 2013 and was tested on each of the test instances described at the start of subsection 3.3. For this implementation $p = 100$ and $p' = 30$ which implies that 30% of each generation is comprised of the best offspring from all of parents from the prior generation with the remainder being the best parents from the prior generation. Additionally, $I = 1000$ which implies that the EWGA will terminate after an improved feasible offspring is not found after 1,000 consecutive generations. For each test instance, the EWGA was conducted 100 times to avoid initialization bias. These values were employed because they performed well in preliminary testing and they provided some computational parity with the other heuristics. This parity is demonstrated in subsection 3.4.

Finally, a comparison methodology was developed which is a Genetic Algorithm that assumed $r_j = 0.50$ for each gene. Hence, this represents traditional GAs which start with completely random individuals at the start of the procedure and assume that each gene has an equal likelihood to mutate throughout the entire algorithm. This procedure will hereafter be referred to as the ‘Standard GA’ and is subject to the same parameters as the EWGA except for the differences in r_j for each $j \in N$.

The results from these tests are shown in Table 5. For each of the 100 GA procedures conducted on each test instance, the best feasible solution was recorded. The ratio of this value over the base case objective value was calculated and the average of these ratios is shown in Table 5. Additionally, the maximum feasible solution over all of the 100 GA procedures was identified and the ratio of this value over the base case objective value is given in the parentheses. Also provided in Table 5 is the average percentage of the GA tests which found a feasible solution for each test case along with the value of M in (3-16) used for that test. The values of M were determined through experimentation and are a function of n, m, and q. Finally, the last column in Table 5 shows the average ratio of the computational time required to solve the EWGA over the computational time to solve the uniform weighted mutation rate GA.

Table 5. EWGA solution objective and time results

(n, m, q)	M	EWGA Ave Obj Ratio (Max Obj Ratio)	Feas. Ratio	Standard GA Ave Obj Ratio (Max Obj Ratio)	Feas. Ratio	EWGA over Stand. Time Ratio
(250, 5, 5)	3000	0.983 (0.993)	1.000	0.948 (0.978)	0.984	0.624
(250, 10, 10)	6000	0.982 (0.995)	0.963	0.941 (0.983)	0.781	0.823
(250, 25, 25)	15000	0.942 (0.973)	0.070	0.833 (0.851)	0.022	0.901
(500, 5, 5)	6000	0.990 (0.995)	1.000	0.946 (0.971)	1.000	0.521
(500, 10, 10)	12000	0.990 (0.996)	0.985	0.949 (0.981)	0.696	0.638
(500, 25, 25)	30000	0.965 (0.987)	0.204	0.850 (0.884)	0.046	0.792
(1000, 5, 5)	12000	0.993 (0.997)	1.000	0.934 (0.958)	1.000	0.401
(1000, 10, 10)	24000	0.993 (0.997)	0.997	0.941 (0.976)	0.631	0.576

The results in Table 5 clearly demonstrate that across all measures and instances, the EWGA methodology is preferred in comparison to the Standard GA. With respect to solution quality, initializing the GA with the linear solution and making mutation less likely for those variables whose efficiency measures most deviate from one clearly improves the solution quality. As this is the first instance of such a mutation procedure, it is hypothesized that similar results would also occur for other KP variants, but additional research is needed to test such a theory. With respect to solution time, the EWGA again performs significantly better. These results demonstrate that the EWGA is able to find better, feasible solutions in a shorter amount of time than the Standard GA.

With respect to the different problem instances, the EWGA clearly performs better for less constrained problems. This result is hypothesized to be a result of the EWGA not featuring a repair operation which can turn infeasible offspring into feasible MDMKP solutions. While such operators are present in GAs applied to KPs and MKPs, there is no simple mechanism to guarantee feasible MDMKP solutions. Hence, the EWGA is not recommended in highly constrained instances. However, assuming the number of constraints is held constant, the results in Table 5 show that EWGA performance increases as the problem size grows. Future research is therefore recommended to test if larger instances continue to result in better performance for the EWGA.

3.4 Discussion

The discussion of each technique in subsection 3.3 solely compared the computational results against the test options within the algorithm and against the base case (solving the problems solely with CPLEX). However, this discussion did not compare the results against one another or against other developed MDMKP solution methods. The purpose of this section is to present and discuss such results as well as to discuss future research opportunities for each of the developed techniques.

In addition to using CPLEX as a comparison solution methodology, the ACT method developed by Hvattum et al. (2010) was employed to solve each test instance. As previously stated, the subproblem solver used in this ACT implementation was CPLEX which is reported to be outperformed by a combination of CPLEX and Scatter Search. CPLEX alone was chosen for this implementation as it is easier to implement and provided equivalent or only slightly worse results than the problems solved with both methodologies. In this implementation, all parameters were kept the same as those recommended by Hvattum et al. and a total of eight hours was given as the maximum processing time. Note that it is also possible for the ACT algorithm to terminate early if the linear problem (which has cuts continually added to it by the suproblem) returns an answer less than best feasible solution identified thus far.

The results in Table 6 through Table 8 compare the ACT solutions against the techniques described in subsection 3.3. Note that the Standard GA is omitted in each table since it was dominated by the EWGA for each test instance. Furthermore, the results for the Kernel Search assume each bucket size was tested for each instance and the final solution was selected as the maximum value over all of the buckets as the results from Table 4 demonstrate there is no ideal bucket strategy. Each column in Table 6 through Table 8 lists a different solution method and each row shows a different test combination. The values presented in Table 6 are the average objective value ratios for the efficiency measure techniques over the ACT methodology while the values in Table 7 are the average solution time ratios for the efficiency measure techniques over the ACT solution method. The values in Table 8 show the count of instances in which the solution method identified the best known solution. In the case where multiple techniques identified the best solution, they are both included in Table 8. Shown in parenthesis in Table 8 is the count of instances in which the ACT solution method identified a feasible solution when the indicated efficiency measure based solution method could not. Those instances where they both identified the same number of feasible solutions are omitted.

Table 6. Average ratio of the efficiency measures based solution values over the ACT solution values

(n, m, q)	EWGA	Kernel		Fixed-Core						
		Unif.	Exp.	δ_A	δ_B	δ_C	δ_D	δ_E	δ_F	δ_G
(250, 5, 5)	0.993	1.000	0.999	0.999	1.000	1.000	0.999	1.000	0.999	1.000
(250, 10, 10)	0.994	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
(250, 25, 25)	0.980	0.997	0.987	1.008	1.013	1.020	1.019	1.020	1.011	1.018
(500, 5, 5)	0.995	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
(500, 10, 10)	0.996	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
(500, 25, 25)	0.985	1.003	1.006	1.008	1.007	1.008	1.009	1.007	1.008	1.007
(1000, 5, 5)	0.996	1.000	1.000	0.999	1.000	0.999	0.999	1.000	0.999	1.000
(1000, 10, 10)	0.997	1.000	1.000	0.999	0.999	1.000	1.000	1.000	1.000	1.000

Table 7. Average ratio of the efficiency measures based solution times over the ACT solution times

(n, m, q)	EWGA	Kernel		Fixed-Core						
		Unif.	Exp.	δ_A	δ_B	δ_C	δ_D	δ_E	δ_F	δ_G
(250, 5, 5)	0.056	0.042	0.016	0.006	0.018	0.507	0.009	0.636	0.009	0.415
(250, 10, 10)	0.265	0.083	0.030	0.014	0.097	0.108	0.017	0.139	0.024	0.148
(250, 25, 25)	0.044	0.183	0.111	0.995	0.995	0.995	0.995	0.995	0.995	0.995
(500, 5, 5)	0.087	0.189	0.041	0.334	0.479	0.512	0.290	0.467	0.190	0.640
(500, 10, 10)	0.086	0.241	0.109	0.419	0.502	0.568	0.382	0.612	0.477	0.616
(500, 25, 25)	0.040	0.291	0.239	0.996	0.996	0.996	0.996	0.996	0.997	0.996
(1000, 5, 5)	0.156	0.289	0.186	0.001	0.002	0.001	0.001	0.003	0.001	0.002
(1000, 10, 10)	0.161	0.294	0.221	0.002	0.016	0.035	0.002	0.005	0.007	0.041

Table 8. Count of instances each solution method identified the best solution (Count of instances not solved by efficiency based methods which were solved by ACT if greater than 0 in parentheses)

(n, m, q)	ACT	EWGA	Kernel		Fixed-Core						
			Unif.	Exp.	δ_A	δ_B	δ_C	δ_D	δ_E	δ_F	δ_G
(250, 5, 5)	9	0	6	3	3	6	7	3	7	4	7
(250, 10, 10)	9	0	9	7	7	7	9	6	8	8	8
(250, 25, 25)	2	2 (1)	1	1	0 (7)	1 (5)	1 (4)	0 (3)	4 (1)	0 (5)	1 (1)
(500, 5, 5)	8	0	7	6	5	2	4	6	6	6	2
(500, 10, 10)	2	0	2	2	1	4	1	4	3	2	1
(500, 25, 25)	0	0 (2)	0	0	0 (1)	1	5	3 (2)	0	1	0
(1000, 5, 5)	4	0	5	3	0	0	1	0	0	0	0
(1000, 10, 10)	6	0	3	2	0	0	0	0	0	0	0

First, one of the key results from Table 6 and Table 8 is that the EWGA is not competitive when compared to the other solution methods. While this may be partly explained by the non-time based stopping criteria, the results in Table 7 show that the EWGA had solution times where were on the same scale with the other methods. It is hypothesized that the EWGA (and Genetic Algorithms in general) are not well suited for solving MDMKP instances due to the lack of a repair operation. As previously stated, this operation was not included as there is no easy procedure which can guarantee feasibility for the generic MDMKP. Hence, this often results in instances in which long stretches of populations have no feasible individuals. The EWGA is therefore not recommended for future research unless a suitable repair operation is developed.

With respect to all of the other solution methods from subsection 3.3, the results in Table 6 demonstrate that the efficiency measure based solution methods are highly competitive on average compared with the ACT solution methodology, but the recommended solution method differs based on the test instance type. For example, the ACT generally performs better on average compared with the other solution methods for the $(250{,}5{,}5)$ instances. Note that the Kernel Search with uniformly sized buckets and some of the fixed core sizes are equally as competitive in these instances. Otherwise, the most constrained problem (e.g. those with 50 total knapsack and demand constraints) demonstrate the advantages of the efficiency-based techniques compared with the ACT procedure. During these instances, all fixed cores sizes outperform the ACT method for $n = 250$ and all fixed core sizes and both Kernel Search techniques outperform the ACT method for $n = 500$. Hence, the results in Table 6 appear to favor the ACT method for simpler, less constrained problems, but larger and more constrained problems are better solved using either the Kernel Search or Fixed-Core methods.

The results from Table 8 further confirm these findings. With respect to smaller and simpler problems, the ACT method is the superior solution method as a majority of the highest quality solutions are a result of this methodology. However, it should be noted that Fixed-Core and Kernel Search methods also find a high percentage of the same solutions for these instances. With respect to the larger and most constrained problems, the ACT does not perform as well. In these instances, the Fixed-Core tests often are the sole solution methods which find the highest quality solutions, even in comparison to the Kernel Search methods which did report promising findings in Table 6. The clear disadvantage of the Fixed-Core methods is that they frequently result in infeasible solutions in these instances, especially compared with the ACT procedure and the Kernel Search procedures, as observed by the values in parentheses.

Finally, the results from Table 7 demonstrate the true advantages of the efficiency measure based solution methods. For nearly all of the solution methods and test instances, the efficiency measure based solution techniques terminated significantly faster on average than the ACT solution methodology. This is especially true for the Kernel Search method which reported high quality average solutions in typically 20% or less of the computational time. While this technique did not frequently determine the best solution as shown in Table 8, it is highly recommended if solution time is of major importance. In comparison, the Fixed-Core solution method frequently solved problems much faster for all but the most constrained instances in which both the ACT method and the Fixed-Core solution methods needed the complete allotted time.

In summary, the developed efficiency measures can be applied to numerous solution methods such that MDMKP test instances can be solved in an efficient manner. Three such solution methods were created and compared with a commercial solver and an existing MDMKP solution heuristic. The results demonstrated that for simpler and less constrained problems, the ACT solution method is only slightly preferred with respect to solution quality, but is greatly outperformed with respect to solution time by the Fixed-Core and Kernel Search methods. Therefore the ACT method is recommended only if solution time is not a factor. For the larger problems, the best technique is to use the Fixed-Core methodology, but this may result in feasibility issues. Hence, it is recommended that the Fixed-Core be tested first and if a feasible solution is not identified quickly, the ACT procedure can be performed. If time is a factor in such solution methods, than the Kernel Search method should be substituted for the ACT procedure.

In regards to the methodologies themselves, the EWGA technique is not currently recommended unless future research can identify a reliable repair operator for infeasible solutions or practitioners do not have access to commercial software. For the Kernel Search, the exponential approach is recommended for constructing buckets unless slight improvements in solution quality are more important than drastic time savings. At this moment, no strategy is recommended for the number of buckets to employ other than to test a range of buckets and retain the best solution from those tests. Future research for the Kernel Search could be conducted to make the procedure iterative as similarly completed by Angelelli, Mansini and Speranza (2010). Also, it could be possible to solve the Kernel Search subproblems with solution methods other than CPLEX to observe the impact on both solution quality and time. Finally, the Fixed-Core technique is highly recommended to solve MDMKPs heuristically due to its ease of implementation and high quality results for most problems. The recommended core size based on this research is $\delta_B = 0.15n$ due to the results shown in Table 1, Table 6, and Table 8, but other problem instances may find other core sizes provide better results. Ultimately, it is recommended that a small core size is first tested and if the results are not satisfactory, the core size is increased. Future research into the Fixed-Core technique includes testing new core sizes as well as testing either exact or heuristic solvers in lieu of using CPLEX.

3.5 Conclusion

With respect to the mobile retailer product mix problem, the best MDMKP heuristic for implementation depends on the mobile retailer. For instance, the EWGA heuristic is the sole technique which does not require commercial software so it can be used by any practitioner even though it performed the worst in comparison to the other techniques. By comparison, the Kernel Search and Fixed-Core heuristics performed the best with respect to solution quality, but both of these techniques require commercial solvers which are unlikely to be available to most mobile retailers. Therefore, mobile retailers who seek to have access to the best techniques are recommended to partner with local research institutions and universities to assist with their operational planning. If such partnerships are not possible, the EWGA is the recommended approach to addressing their product mix decision. If enough mobile retailers were to implement this approach, future research should be performed to identify a repair operator which is expected to drastically improve the EWGA technique.

The solution times for these algorithms should also be addressed with respect to the mobile retailer product mix problem. As discussed, each of the solution algorithms frequently needed upwards of eight hours to solve the tested MDMKP instances. This was especially true for instances with 1000 variables and 20 combined constraints. This is equivalent to a mobile retailer who is choosing their ideal product mix among 1000 different items and they have 20 or more requirements or restrictions on this mix. In these cases, the retailer is not recommended to continually change and resolve their product mix due to the long solution times. However, such retailers are likely the exception as most mobile retailers will not have a high quantity of possible items to stock nor will they have that many requirements or limitations. In these cases, the mobile retailer can continually modify their product mix on a weekly basis to take advantage of fluctuations in item costs from their suppliers. This system will not only benefit the retailer by increasing their profit margins, but it may also benefit customers as the product variability will maintain customer interest in the retailer so long as some staple goods are continuously stocked.

Regardless of the chosen heuristic, each of the developed solutions are applicable to the generic mobile retailer product mix decision. These problems are referred to as generic as any product mix decision can be modeled by an MDMKP assuming that the objective and all constraints can be expressed as linear functions. The advantage of this formulation is it permits a wide array of scenarios with respect to the objectives and constraints. For instance, one retailer may wish the profit of the product mix to be expressed as a constraint while another may wish it to be an objective. Regardless of these differences, a mobile retailer can use the MDMKP to increase the profitability or nutritional quality of their product mix or they can use the MDMKP to decrease the consumer cost of their product mix. An example of how the MDMKP formulation and the developed solution algorithms from this chapter can be used to accomplish all three of these improvements is demonstrated in subsection 7.3 through a case study using operational data from a mobile retailer.

This chapter was included in Wishon and Villalobos (2016b).

CHAPTER 4

THE TWO CONSTRAINT MOBILE RETAILER PRODUCT MIX PROBLEM

Within this chapter, the solution algorithm for the two constraint mobile retailer product mix problem will be discussed which is modeled as a DKP. For this version of the problem, it is assumed that the product mix of a mobile retailer can be modeled with a single linear objective function, one linear knapsack constraint, one linear demand constraint, and decision variables which only determine whether or not to include a grocery item on the retailer (the quantity to stock is predetermined). While these assumptions are more restrictive than the type of product mix decisions discussed in Chapter 3, the described scenario is still common within mobile retailers. For instance, a mobile retailer may require a stocked mix to earn a maximum revenue, not exceed the space on the retailer, and meet a minimum nutritional content or, for another example, a retailer may want the stocked mix to be as affordable as possible for its customers, not exceed the space on the retailer, and meet a minimum profit margin. These scenarios, and many more combinations, can all be modeled as a DKP and a dedicated solution algorithm for these problems has been developed.

The motivation for creating a specialized algorithm for the DKP is two-fold. First and foremost, a solution algorithm developed specifically for the DKP will be able to develop higher quality solutions in less time since a dedicated DKP solver should be able to exploit the simplicity of DKP formulation. Secondly, the best MDMKP heuristics from Chapter 3 required the use of commercial solvers. The simplicity of the DKP formulation permits high quality, exact algorithms which do not required commercial software.

The first exact solution algorithm for the DKP is presented in this chapter and is hereafter called DKPSOLVE. Multiple types of tests instances were developed and solved with both DKPSOLVE and commercial solvers. This chapter concludes with a discussion of the computational results between the two solution methods (commercial software versus DKPSOLVE) as well as a discussion regarding the applicability of the solver for mobile retailers.

4.1 The DKP and DKP relaxations

To formulate the DKP, assume there are n items such that each item $i \in \{1, \dots, n\}$ is defined by a profit p_i, knapsack weight w_i, and demand weight v_i. Assume the knapsack limit is denoted by C and the demand requirement is denoted by R. The objective of the DKP is to select a subset of items such that the total profit is as large as possible while the total knapsack summation does not exceed C and the total demand summation is at least R. Hence the demand-constrained KP is formulated as the following binary problem

(DKP) Maximize: $\sum_{i=1}^{n} p_i x_i,$ (4-1)

$\sum_{i=1}^{n} w_i x_i \leq C,$ (4-2)

$\sum_{i=1}^{n} v_i x_i \geq R,$ (4-3)

$x_i \in \{0,1\} \quad (i = 1, \dots, n),$ (4-4)

where x_i takes the value 1 if and only if item i is selected for inclusion.

Unlike traditional KP problems, the DKP can have looser assumptions with respect to the coefficients. For an item i, p_i is only assumed to be integer while w_i and v_i are nonnegative integers. Negative profit coefficients are permitted since inclusion of such an item may be necessary to satisfy the demand constraint. In addition, it is assumed that the knapsack and revenue thresholds are positive integers as well as

$$\sum_{i=1}^{n} w_i \geq C, \tag{4-5}$$

$$\sum_{i=1}^{n} v_i \geq R, \tag{4-6}$$

and $\max_i(w_i) \leq C.$ (4-7)

Knapsack and revenue thresholds are assumed to be positive integers since negative values would cause infeasibility and zero values could either create an infeasible problem or one which could be solved using existing binary KP techniques. If (4-5) is violated, the knapsack constraint can be removed and the problem can be solved using binary KP solution techniques as documented in subsection 4.1.1. If (4-6) is violated, the problem is infeasible. If (4-7) is violated, the associated item can be removed from consideration in the problem.

Solving the DKP requires the use of the Lagrangian, surrogate, and continuous relaxations. These will be employed throughout the solution algorithm to obtain strong upper bounds. For the following, let P be a given problem and let $z(P)$ represent its optimal solution value.

4.1.1 DKP Lagrangian relaxations

There are three possible Lagrangian relaxations for the DKP. The Lagrangian relaxation of demand constraint (4-3) using the nonnegative Lagrangian multiplier λ is formulated as follows:

$L_R(DKP, \lambda)$ Maximize: $-\lambda R + \sum_{i=1}^{n}(p_i + \lambda v_i)x_i,$ (4-8)

$$\sum_{i=1}^{n} w_i x_i \leq C, \tag{4-9}$$

$x_i \in \{0,1\} \quad (i = 1, \dots, n).$ (4-10)

$L_R(DKP, \lambda)$ is a binary KP with unrestricted objective coefficients. While not solvable in polynomial time, it can be solved quickly in most practical instances as demonstrated by Martello, Pisinger, and Toth (1999). For any nonnegative multiplier λ, $L_R(DKP, \lambda)$ provides the upper bound

$$LU_R(\lambda) = \lfloor z(L_R(DKP, \lambda)) \rfloor$$

for the DKP where the floor function is a result of the non-integrality of λ. Hence the tightest upper bound based on $L_R(DKP, \lambda)$ is

$LU_R(\lambda^*) = \min_{\lambda \geq 0} LU_R(\lambda).$ (4-11)

Similarly, the Lagrangian relaxation of the knapsack constraint (4-2) using the nonnegative multiplier μ, is formulated as follows:

$L_C(DKP, \mu)$ Maximize: $\mu C + \sum_{i=1}^{n} (p_i - \mu w_i) x_i,$ (4-12)

$\sum_{i=1}^{n} v_i x_i \geq R,$ (4-13)

$x_i \in \{0,1\} \quad (i = 1, \dots, n).$ (4-14)

While $L_C(DKP, \mu)$ does not have a knapsack constraint, the relaxation can be solved as a standard, binary KP. To demonstrate this property, consider the disjoint sets of the variables I' and I'' where $i \in I'$ if $p_i - \mu w_i \geq 0$ and $i \in I''$ otherwise. Thus, solving $L_C(DKP, \mu)$ is equivalent to solving the following binary problem:

$\overline{L_C}(DKP, \mu)$ $\sum_{i=1}^{n} (p_i - \mu w_i) + \mu C + \max[\sum_{i \in I''} -(p_i - \mu w_i) x_i],$ (4-15)

$\sum_{i \in I''} v_i x_i \leq (\sum_{i \in I'} v_i + \sum_{i \in I''} v_i) - R,$ (4-16)

$x_i \in \{0,1\} \quad (i \in I''),$ (4-17) where $z\big(L_C(DKP,\mu)\big) = z\big(\overline{L_C}(DKP,\mu)\big)$. An item i is included in the optimal solution to $L_C(DKP,\mu)$ if $p_i - \mu w_i \geq 0$ or it is excluded from the optimal solution of $\overline{L_C}(DKP,\mu)$. This equivalence is based on two cases: $\sum_{i\in I'} v_i \geq R$ and $\sum_{i\in I'} v_i < R$. For the former, (4-16) will permit all $i \in I''$ to be placed in the knapsack which is equivalent to only selecting the positive profit items for $L_C(DKP,\mu)$. For the later, $\overline{L_C}(DKP,\mu)$ attempts to find the most negative subset of items $i \in I''$ to place in the knapsack such that those left over are the least negative subset of items whose demand sum must exceed $R - \sum_{i\in I'} v_i$.

Given a solution to $L_C(DKP,\mu)$, an upper bound on the DKP for any nonnegative multiplier μ is

$$LU_C(\mu) = \big\lfloor z\big(L_C(DKP,\mu)\big)\big\rfloor$$

where the floor function is a result of the non-integrality of μ. The tightest upper bound based on $L_C(DKP,\mu)$ is therefore

$$LU_C(\mu^*) = \min_{\mu\geq 0} LU_C(\mu). \tag{4-18}$$

In addition to these two Lagrangian relaxations, it is also possible to relax both constraints (4-2) and (4-3) with nonnegative multipliers $\hat{\mu}$ and $\hat{\lambda}$ respectively. This results in the problem $L_{C,R}(DKP,\hat{\mu},\hat{\lambda})$ defined by

$$\hat{\mu}C - \hat{\lambda}R + \max \textstyle\sum_{i=1}^{n}\big(p_i - \hat{\mu}w_i + \hat{\lambda}v_i\big)x_i, \tag{4-19}$$

subject only to (4-4). This problem is easily solved in $O(n)$ time since an item i is selected only if $p_i - \hat{\mu}w_i + \hat{\lambda}v_i \geq 0$, even if constraint set (4-4) is relaxed to its linear equivalent. Hence, $L_{C,R}(DKP,\hat{\mu},\hat{\lambda})$ has the integrality property so the floor of the minimum value for this relaxation, $LU_{C,R}\big(\hat{\mu}^*,\hat{\lambda}^*\big)$, is equivalent to the floor of the continuous relaxation of the DKP. Furthermore, $LU_{C,R}\left(\hat{\mu}^*, \hat{\lambda}^*\right)$ is a looser upper bound than (4-11) and (4-18) since it is a relaxation of both Lagrangian relaxations.

4.1.2 DKP surrogate relaxation

Similar to the two-constraint KP from Martello and Toth (2003), the surrogate relaxation can also provide an upper bound for the DKP. Consider two nonnegative multipliers α and β. The surrogate relaxation $S(DKP, \alpha, \beta)$ is formulated as:

$$S(DKP, \alpha, \beta) \text{ Maximize: } \sum_{i=1}^{n} p_i x_i, \tag{4-20}$$

$$\sum_{i=1}^{n} \left(\alpha w_i - \beta v_j\right) x_i \leq \alpha C - \beta R, \tag{4-21}$$

$$x_i \in \{0,1\} \quad (i = 1, \dots, n). \tag{4-22}$$

Similar to the Lagrangian relaxations, $S(DKP, \alpha, \beta)$ is a binary KP with possibly negative volumes and capacity. After preprocessing, $S(DKP, \alpha, \beta)$ can be solved as a binary KP such that the following upper bound is obtained:

$$SU(\alpha, \beta) = z\left(S(DKP, \alpha, \beta)\right).$$

Hence the best upper bound is

$$SU(\alpha^*, \beta^*) = \min_{\alpha, \beta \geq 0} SU(\alpha, \beta). \tag{4-23}$$

It should be noted that there is no dominating relationship between upper bounds (4-11), (4-18), and (4-23).

4.1.3 DKP continuous relaxations

The continuous relaxations can also provide an upper bound on the DKP. For notation, consider any problem P and let $C(P)$ represent the continuous relaxation of that problem. The continuous relaxation of the DKP, $C(DKP)$, is formulated as the linear program of (4-1) - (4-3) and $0 \leq x_i \leq 1$ for all $i \in \{1, \ldots, n\}$. This provides the following upper bound for the DKP

$$CU = \lfloor z(C(DKP)) \rfloor. \tag{4-24}$$

Given this bound, four separate upper bounds now exist for the DKP: (4-11), (4-18), (4-23), and (4-24). Note, $LU_{C,R}(\hat{\mu}^*, \hat{\lambda}^*)$ is not a unique upper bound as it is equal to CU as stated previously.

In addition to these bounds, the continuous relaxations of Lagrangian and surrogate relaxations will be computationally important for solving the DKP. Specifically, define the two continuous Lagrangian relaxation upper bounds as $CLU_R(\tilde{\lambda}) = \min_{\lambda \geq 0} \left\lfloor z\left(C\left(L_R(DKP, \lambda)\right)\right) \right\rfloor$ and $CLU_C(\tilde{\mu}) = \min_{\mu \geq 0} \left\lfloor z\left(C\left(L_C(DKP, \mu)\right)\right) \right\rfloor$ and define the continuous surrogate relaxation upper bound as $CSU(\tilde{\alpha}, \tilde{\beta}) = \min_{\alpha, \beta \geq 0} \left\lfloor z\left(C\left(S(DKP, \alpha, \beta)\right)\right) \right\rfloor$. As similarly demonstrated by Martello and Toth (2003) for the two-constraint KP, the DKP has the following relationship between the continuous upper bounds:

$$CU = CLU_R(\tilde{\lambda}) = CLU_C(\tilde{\mu}) = CSU(\tilde{\alpha}, \tilde{\beta}). \tag{4-25}$$

To demonstrate (4-25), the proof for $CU = CLU_C(\tilde{\mu})$ is provided. The remaining equalities are then immediately obtained using the same reasoning. First, for any nonnegative λ and μ, $z(C(DKP)) \leq z\left(C\left(L_C(DKP, \mu)\right)\right)$ since $C(L_C(DKP, \mu))$ is a relaxation of $C(DKP)$. Secondly, because the feasible region of $C(L_C(DKP, \mu))$ is a subset of $C\left(L_{C,R}(DKP, \mu, \lambda)\right)$ and (4-12) is less than (4-19) for any solution satisfying (4-13), then:

$$CU = LU_{C,R}\left(\hat{\mu}^*, \hat{\lambda}^*\right) \geq CLU_C(\hat{\mu}^*) \geq CLU_C(\tilde{\mu}) \geq CU.$$

Hence $CU = CLU_C(\tilde{\mu})$ with the remainder of (4-25) being proven through similar reasoning.

4.2 DKP upper bounds

The purpose of the upper bounds presented in this section is to provide tight limits on the optimal binary solution during both the Reduction and Expanding Core phases of the solution procedure (introduced in subsections 4.3 and 4.4 of this chapter). Since these upper bounds may have to be calculated multiple times during the Expanding Core procedure, only the strongest upper bounds should be used and they must be computed efficiently. As demonstrated by (4-25), all of the continuous relaxations and $L_{C,R}(DKP, \hat{\mu}^*, \hat{\lambda}^*)$ provide the same bound. Hence, only (4-11), (4-18), and (4-23) need to be computed since no dominance exists between these bounds and they are all tighter than the continuous relaxation. The disadvantage of these bounds is that optimally solving the multipliers for these problems is computationally expensive. Instead, it is recommended that the optimal $\tilde{\lambda}, \tilde{\mu}, \tilde{\alpha},$ and $\tilde{\beta}$ from the continuous relaxations of these bounds be used as substitutes since it is expected these values will still provide a high quality upper bound. The remainder of this section demonstrates how to solve for these values in polynomial time.

4.2.1 Optimal DKP dual values

The analysis which follows demonstrates that only one of the continuous relaxations must be solved, specifically $CLU_R(\tilde{\lambda})$, such that all of the continuous multipliers can be calculated. Furthermore, a $O(n^2)$ algorithm is presented to find $\tilde{\lambda}$. While this is not as fast as the multidimensional search technique presented by Megiddo and Tamir (1993), the implemented approach, motivated by the similar technique demonstrated by Martello and Toth (2003) for their two-constraint KP, is simpler to implement and provides extremely competitive results as demonstrated in subsection 4.6.

For any given λ, $C\big(L_R(DKP,\lambda)\big)$ is a continuous KP which can be solved by ordering the items such that

$$(p_i + \lambda v_i)/w_i \geq (p_{i+1} + \lambda v_{i+1})/w_{i+1} \qquad (i = 1, \dots, n-1) \tag{4-26}$$

which provides the optimal decision variables values, x_i^*, of

$$\begin{aligned} x_i^* &= 1 \qquad (j = 1, \dots, b(\lambda) - 1), \\ x_{b(\lambda)}^* &= \begin{cases} \left(C - \sum_{i=1}^{b(\lambda)-1} w_i\right)/w_{b(\lambda)} & \text{if } p_i + \lambda v_i \geq 0 \\ 0 & \text{otherwise} \end{cases} \\ x_i^* &= 0 \qquad (j = b(\lambda) + 1, \dots, n), \end{aligned} \tag{4-27}$$

where $b(\lambda)$ is the break item defined by

$$b(\lambda) = \min\left\{j : \textstyle\sum_{i=1}^{j} w_i > C \text{ or } p_i + \lambda v_i < 0\right\}.$$

While sorting the variables according to (4-26) allows for easy computation of (4-27) and the associated solution value, Balas and Zemel (1980) provide a $O(n)$ median finding algorithm which solves the problem without sorting. It is this approach which will be implemented to solve $CLU_R(\tilde{\lambda})$ in $O(n^2)$ time as shown in Theorem 2.

To demonstrate the possible values for $\tilde{\lambda}$, assume that the items are ordered according to (4-26) and ties are broken according to non-increasing demand-to-knapsack constraint coefficient ratios. Furthermore, let $B(\lambda)$ represent a set of items such that $b \in B(\lambda)$ if and only if $(p_b + \lambda v_b)/w_b = \big(p_{b(\lambda)} + \lambda v_{b(\lambda)}\big)/w_{b(\lambda)}$ where $b(\lambda)$ represents the index of the first element in $B(\lambda)$. Hence $B(\lambda)$ is the set of possible break items for a given λ. Finally, let

$$LHS_{MIN}(\lambda) = \sum_{i=1}^{b(\lambda)-1} v_i + \min\left\{\sum_{b\in B(\lambda)} v_b x_b \,\middle|\, \sum_{b\in B(\lambda)} w_b x_b \le C - \sum_{i=1}^{b(\lambda)-1} w_i\,, 0 \le x_b \le 1 \text{ for } b \in B(\lambda)\right\}$$

and

$$LHS_{MAX}(\lambda) = \sum_{i=1}^{b(\lambda)-1} v_i + \max\left\{\sum_{b\in B(\lambda)} v_b x_b \,\middle|\, \sum_{b\in B(\lambda)} w_b x_b \le C - \sum_{i=1}^{b(\lambda)-1} w_i\,, 0 \le x_b \le 1 \text{ for } b \in B(\lambda)\right\}$$

represent the minimum and the maximum possible values respectively for the left-hand side of the relaxed constraint (4-3) in the optimal solution of $C\big(L_R(DKP,\lambda)\big)$. Observe that the corresponding solution implied by $LHS_{MIN}(\lambda)$ and $LHS_{MAX}(\lambda)$ result in the same value for $z\left(C\big(L_R(DKP,\lambda)\big)\right)$ as all $B(\lambda)$ have the same efficiency measure values.

Lemma 1. $LHS_{MIN}(\lambda)$ and $LHS_{MAX}(\lambda)$ are monotonically nondecreasing as λ increases.

Proof: Consider either $LHS_{MIN}(\lambda)$ or $LHS_{MAX}(\lambda)$. Let a and b be two items and assume λ' is strictly less than λ''. Observe that $LHS_{MIN}(\lambda)$ or $LHS_{MAX}(\lambda)$ can only change values if two items were to exchange places in ordering (4-26). Therefore, assume $(p_a + \lambda' v_a)/w_a \ge (p_b + \lambda' v_b)/w_b$ and $(p_a + \lambda'' v_a)/w_a \le (p_b + \lambda'' v_b)/w_b$. By subtracting the second inequality from the first, $v_a(\lambda' - \lambda'')/w_a \ge v_b(\lambda' - \lambda'')/w_b$ which results in $v_a/w_a \le v_b/w_b$.

Observe that as λ increases, the objective coefficient $p_i + \lambda v_i$ can only increase. Hence $\sum_{i=1}^{n} w_i x_i^*$ as defined in (4-27) cannot decrease as λ increases. Since items with a greater demand-to-knapsack constraint coefficients ratios move forward in the ordering as λ increases, then it is clear that $LHS_{MIN}(\lambda'') \geq LHS_{MIN}(\lambda')$ and $LHS_{MAX}(\lambda'') \geq LHS_{MAX}(\lambda')$. ■

Lemma 2. The optimal multiplier for $CLU_R(\tilde{\lambda})$ is one of the following values:

(a) $\tilde{\lambda} = 0$

(b) $\tilde{\lambda} = -p_i/v_i$ if $p_i < 0$ and $v_i > 0$ for $i = 1, \dots, n$

(c) $\tilde{\lambda} = (p_i w_j - p_j w_i)/(v_j w_i - v_i w_j)$ for all $1 \leq i < j \leq n$ such that $\tilde{\lambda} > 0$

Proof: Assume that for any given λ, the optimal solution for $C(L_R(DKP, \lambda))$ has been determined using the median finding algorithm by Balas and Zemel (1980). Three cases are possible:

1) $LHS_{MIN}(\lambda) \leq R$ and $LHS_{MAX}(\lambda) \geq R$: Since the decision variables are continuous, then there exists an optimal solution vector, $\boldsymbol{x}^*$, such that $LHS(\lambda) = R$ where $LHS(\lambda) = \sum_{i=1}^{n} v_i x_i^*$. By complementary slackness, $\lambda = \tilde{\lambda}$.

2) $LHS_{MIN}(\lambda) > R$: Given this case, the objective is to find the minimum multiplier λ' such that $\lambda' < \lambda$ where the optimal solution $\boldsymbol{x}^*$ is the same for both $C(L_R(DKP, \lambda))$ and $C(L_R(DKP, \lambda'))$, but $z\left(C(L_R(DKP, \lambda))\right) \geq z\left(C(L_R(DKP, \lambda'))\right)$. Since the solution vectors defining $LHS_{MIN}(\lambda)$ and $LHS_{MAX}(\lambda)$ simply represent alternate optima with respect to $C(L_R(DKP, \lambda))$, the general definition of $LHS(\lambda)$ will be employed to represent any of these alternate options.

For the first case, consider the situation where $p_b + \lambda v_b$ is nonnegative for all $b \in B(\lambda)$. λ and λ' will provide the same optimal solution vector if two conditions are met. First, the objective coefficient for the break items remains nonnegative. The second is that λ' provides the same partition as λ when the variables are grouped based on their efficiency measures compared to the break item. Therefore, λ and λ' are the same assuming the following conditions are met:

$$p_b + \lambda' v_b \geq 0 \quad b \in B(\lambda), \tag{4-28}$$

$$(p_i + \lambda' v_i)/w_i \geq \left(p_{b(\lambda)} + \lambda' v_{b(\lambda)}\right)/w_{b(\lambda)} \quad j = 1, \dots, b(\lambda) - 1, \tag{4-29}$$

$$(p_i + \lambda' v_i)/w_i \leq \left(p_{b(\lambda)} + \lambda' v_{b(\lambda)}\right)/w_{b(\lambda)} \quad j = b(\lambda) + |B(\lambda)|, \dots, n. \tag{4-30}$$

Since λ' and λ result in the same partition, then the objective function for $C\left(L_R(DKP, \lambda')\right)$ can be rewritten as $\sum_{i=1}^{n} p_i x_i^* - \lambda'\left(R - LHS(\lambda')\right)$. Since λ' is selected such that $LHS(\lambda) = LHS(\lambda')$, then the minimum λ' satisfying (4-28) - (4-30) is desired. The minimum λ' satisfying criteria (4-29) and (4-30) is $\overline{\lambda'} = \max\left\{\left(p_i w_{b(\lambda)} - p_{b(\lambda)} w_i\right)/\left(v_{b(\lambda)} w_i - v_i w_{b(\lambda)}\right)\right\}$ for any $i < b(\lambda)$ where $v_i/w_i > v_{b(\lambda)}/w_{b(\lambda)}$ or $i > b(\lambda) + |B(\lambda)|$ where $v_i/w_i < v_{b(\lambda)}/w_{b(\lambda)}$. To demonstrate the conditions for $\overline{\lambda'}$, observe that (4-29) is satisfied automatically in the case where $v_{b(\lambda)}/w_{b(\lambda)} \geq v_i/w_i$ using a similar observation from the proof of Lemma 1 that two items will only swap places due to a change in λ based on their demand-to-knapsack ratios. Similar reasoning applies to (4-30) in the case where $v_{b(\lambda)}/w_{b(\lambda)} \leq v_i/w_i$. Finally, the minimum λ' satisfying (4-28) - (4-30) can be calculated as

$$\lambda' = \max\left\{\overline{\lambda'}, \left(-p_{b(\lambda)}/v_{b(\lambda)} \,\middle|\, p_{b(\lambda)} < 0 \text{ and } v_{b(\lambda)} > 0\right)\right\}.$$

In the case where $p_b + \lambda v_b$ is negative for all $b \in B(\lambda)$, it is sufficient to impose that none of the objective coefficients changes sign based on the solution (4-27). Hence, the best (i.e. minimum) λ' is $\max\{-p_i/v_i \mid i > b(\lambda) \text{ and } v_i > 0\}$.

3) $LHS_{MAX}(\lambda) < R$: Given this case, the objective is to find the maximum multiplier λ' such that $\lambda' > \lambda$ where the optimal solution $\boldsymbol{x}^*$ is the same for both $C\left(L_R(DKP, \lambda)\right)$

and $C\big(L_R(DKP,\lambda')\big)$, but $z\left(C\big(L_R(DKP,\lambda)\big)\right) \geq z\left(C\big(L_R(DKP,\lambda')\big)\right)$. When $p_b + \lambda v_b$ is nonnegative for all $b \in B(\lambda)$, similar reasoning as case 2 only imposes bounds (4-29) and (4-30) since increasing λ can't create the left hand side of (4-28) to become negative. This provides the following rule where $\overline{\lambda'} = \min\{\big(p_i w_{b(\lambda)} - p_{b(\lambda)} w_i\big)/\big(v_{b(\lambda)} w_i - v_i w_{b(\lambda)}\big)\}$ for any $i < b(\lambda)$ where $v_i/w_i < v_{b(\lambda)}/w_{b(\lambda)}$ or $i > b(\lambda) + |B(\lambda)|$ where $v_i/w_i > v_{b(\lambda)}/w_{b(\lambda)}$.

When $p_b + \lambda v_b$ is negative for all $b \in B(\lambda)$, the only consideration is that the objective coefficient for any $i < b(\lambda)$ does not change sign. Hence, $\lambda' = \min\{-p_i/v_i \,|i < b(\lambda)$ and $v_i > 0\}$.

These three cases demonstrate that the optimal $\tilde{\lambda}$ is one out of a set of finite, calculable values. Specifically, if $LHS_{MIN}(0) \geq R$, then $\tilde{\lambda} = 0$ is the clear dominating solution. Otherwise $\tilde{\lambda} = -p_i/v_i$ assuming $p_i < 0$ and $v_i > 0$ or $\tilde{\lambda} = \big(p_i w_j - p_j w_i\big)/\big(v_j w_i - v_i w_j\big)$ assuming $1 \leq i < j \leq n$ such that $\tilde{\lambda} > 0$ where the restriction $i < j$ is a result of the calculation remaining unchanged if i and j are switched.

■

Solving $C\big(L_R(DKP,\lambda)\big)$ for all of the possibilities listed in Lemma 2 provides a $O(n^3)$ time solution method. However, employing a median finding technique similar to Balas and Zemel (1980) results in the relaxation being solved in $O(n^2)$ time. This algorithm is motivated by the similar approach of Martello and Toth (2003).

Theorem 2. The optimal multiplier $\tilde{\lambda}$ can be determined in $O(n^2)$ time.

Proof: Consider the following procedure titled CDKPSOLVE. The purpose of CDKPSOLVE is to optimally solve $CLU_R(\tilde{\lambda})$, thereby calculating CU and the optimal multiplier $\tilde{\lambda}$. For input, CDKPSOLVE requires n, C, R, $\boldsymbol{p} = \{p_1, p_2, \dots, p_n\}$, $\boldsymbol{w} = \{w_1, w_2, \dots, w_n\}$, and $\boldsymbol{v} = \{v_1, v_2, \dots, v_n\}$ as well as Λ which is the storage for all possible multiplier values from Lemma 2.

```
**procedure** CDKPSOLVE($n, \boldsymbol{p}, \boldsymbol{w}, \boldsymbol{v}, C, R, \Lambda$)
    (a)  Calculate $\Lambda$ if needed
    (b)  Compute $z\left(C\left(L_R(DKP, 0)\right)\right)$, $LHS_{MIN}(0)$, and $LHS_{MAX}(0)$
    (c)  If $LHS_{MIN}(0) \geq R$ or $LHS_{MAX}(0) \geq R$ and $LHS_{MIN}(0) \leq R$ THEN $\tilde{\lambda} = 0$,
         $CLU_R(\tilde{\lambda}) = z\left(C\left(L_R(DKP, 0)\right)\right)$ and STOP
    (d)  Determine median $\lambda$ of $\Lambda$
    (e)  Compute $z\left(C\left(L_R(DKP, \lambda)\right)\right)$, $LHS_{MIN}(\lambda)$, and $LHS_{MAX}(\lambda)$
    (f)  IF $LHS_{MAX}(\lambda) \geq R$ and $LHS_{MIN}(\lambda) \leq R$ THEN $\tilde{\lambda} = \lambda$, $CLU_R(\tilde{\lambda}) =$
         $z\left(C\left(L_R(DKP, \lambda)\right)\right)$ and STOP
    (g)  IF $LHS_{MAX}(\lambda) < R$ THEN remove all values greater than or equal to $\lambda$ from
         $\Lambda$ ELSE remove all values less than or equal to $\lambda$ from $\Lambda$
    (h)  Go to (d)
```

Within CDKPSOLVE, it is assumed that the median finding technique is used in (b) and (e) with which has a complexity of $O(n)$. Furthermore, it is assumed that the technique proposed by Blum et al. (1973) is used to find the median in an unsorted list in (d) which has a complexity of $O(|\Lambda|)$. Note that the guarantee that CDKPSOLVE will terminate with the correct value for $\tilde{\lambda}$ and $CLU_R(\tilde{\lambda})$ is a direct result of Lemma 1 and Lemma 2.

Since (a) through (c) are only performed once, the *total* complexity of these steps is at least $O(n^2)$ due to the number of calculations necessary to determine set Λ according to Lemma 2. To determine the *total* complexity of the remaining steps, note that Lemma 2 demonstrates that $|\Lambda|$ is at most $n(n+1)/2$ and it is halved at each iteration. Hence, the *total* complexity of (d) and (g) is at most bounded by $O(n(n+1)/2 + n(n+1)/4 + n(n+1)/8 + \cdots) = O(n^2)$. This halving also demonstrates that the *total* complexity of (e) is $O(n \log n)$ and the *total* complexity of (f) is $O(\log n)$. Therefore, (a) through (c) are computed once with worst-case complexity of $O(n^2)$ while (d) through (h) have a total worst-case complexity of $O(n^2)$. Hence, CDKPSOLVE can be solved in $O(n^2)$ time. ■

Since this technique only provides one of the optimal continuous multipliers, a similar approach could be developed to determine $\tilde{\mu}, \tilde{\alpha}$, and $\tilde{\beta}$. However, Theorem 3 demonstrates how these values can be calculated based on the solution corresponding to $CLU_R(\tilde{\lambda})$.

Theorem 3. Assume the optimal multiplier and solution corresponding to $CLU_R(\tilde{\lambda})$ are known. The optimal multiplier $\tilde{\mu}$ for minimizing $C(L_C(DKP,\mu))$ is $\tilde{\mu} = \left(p_{b(\tilde{\lambda})} + \tilde{\lambda} v_{b(\tilde{\lambda})}\right)/w_{b(\tilde{\lambda})}$. The optimal multipliers $\tilde{\alpha}$ and $\tilde{\beta}$ for minimizing $C(S(DKP,\alpha,\beta))$ are $\tilde{\alpha} = \tilde{\lambda}$ and $\tilde{\beta} = \tilde{\mu}$.

Proof: Consider the dual formulation of $C(DKP)$ given as

$$\tilde{z} = \min\left(\mu C - \lambda R + \textstyle\sum_{i=1}^{n} \pi_i \left| \begin{matrix} \mu w_i - \lambda v_i + \pi_i \geq p_i \ (i = 1, \dots, n), \\ \lambda, \mu, \pi_i \geq 0 \ (i = 1, \dots, n) \end{matrix} \right.\right). \tag{4-31}$$

In the optimal solution to $C(DKP)$, let I^* represent the items for which $x_i^* = 1$ in the optimal solution. Since the optimal dual values for a continuous problem are equal to the optimal multipliers in a Lagrangian or surrogate relaxations, complimentary slackness of $C(DKP)$ and its dual implies $\tilde{z} = \tilde{\mu}C - \tilde{\lambda}R + \sum_{i \in I^*}(p_i - \tilde{\mu}w_i + \tilde{\lambda}v_i)$ and (4-25) plus the optimality of $C\left(L_R(DKP, \tilde{\lambda})\right)$ implies $\tilde{z} = -\tilde{\lambda}R + \sum_{i \in I^*}(p_i + \tilde{\lambda}v_i) + x^*_{b(\tilde{\lambda})}\left(p_{b(\tilde{\lambda})} + \tilde{\lambda}v_{b(\tilde{\lambda})}\right)$.

Using algebraic manipulation and the value for $x^*_{b(\tilde{\lambda})}$ defined in (4-27), it can be demonstrated that $\tilde{\mu} = \left(p_{b(\tilde{\lambda})} + \tilde{\lambda}v_{b(\tilde{\lambda})}\right)/w_{b(\tilde{\lambda})}$. In regards to implementation, it is important to observe that $\tilde{\mu}$ is the ratio of the objective to constraint coefficients for the break item in $C\left(L_R(DKP, \tilde{\lambda})\right)$ which is easily obtained from the solution procedure outlined by Balas and Zemel (1980).

The proof for the optimal surrogate multipliers is immediate since the optimal dual values for a continuous problem are equal to the optimal multipliers in a Lagrangian or surrogate relaxations. ■

4.2.2 DKP integer relaxations

Using these theorems, the continuous relaxation for DKP can be solved in $O(n^2)$ time which provides the optimal Lagrangian and surrogate multipliers. These multipliers can be used to calculate $\left\lfloor z\left(L_R(DKP, \tilde{\lambda})\right)\right\rfloor$, $\lfloor z(L_C(DKP, \tilde{\mu}))\rfloor$, and $z\left(S(DKP, \tilde{\alpha}, \tilde{\beta})\right)$. Even though this requires the solution of three binary KPs which are NP-hard, solution algorithms such as those by Martello, Pisinger, and Toth (1999) can solve the problems extremely efficiently. The only challenge with this approach is that some of the coefficients in each problem must be scaled to ensure the integrality assumption of Martello, Pisinger, and Toth.

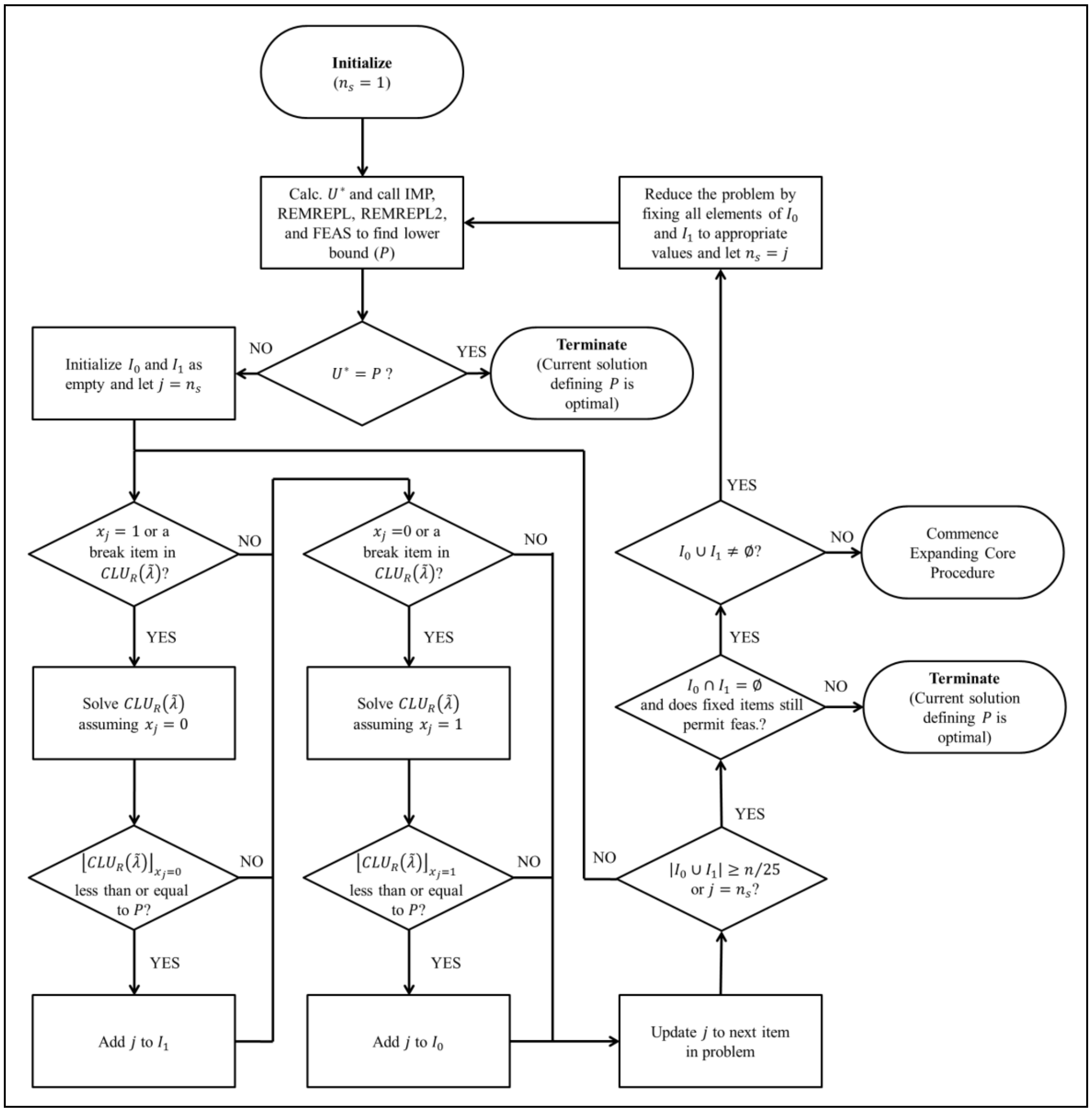
While these values do not guarantee the lowest possible upper bounds (they utilize the optimal multipliers from their continuous counterparts), they provide extremely competitive bounds as demonstrated in the computational experiments in subsection 4.6. Since, no dominance relationship exists between these bounds, all three must be calculated and the best bound is given as

$$U^* = \min\left(\left\lfloor z\left(L_R(DKP, \tilde{\lambda})\right)\right\rfloor, \left\lfloor z\left(L_C(DKP, \tilde{\mu})\right)\right\rfloor, z\left(S\left(DKP, \tilde{\alpha}, \tilde{\beta}\right)\right)\right). \tag{4-32}$$

Implementation of (4-32) in the algorithm that follows also provides a potential starting solution for the DKP. Therefore, ties in the minimum bound are broken first by feasibility of the bound's solution and then by (4-1).

4.3 DKP reduction procedure

The full DKPSOLVE algorithm follows a two phase approach. The first phase, the Reduction procedure, is presented in this section while the second phase, the Expanding Core procedure, is presented in subsection 4.4. The Reduction procedure, hereafter referred to as REDUCE, commences by calculating U^* in (4-32) to determine an upper bound on the current DKP instance. Then four procedures titled IMP, REMREPL, REMREPL2, and FEAS are performed to determine a feasible binary lower bound. These procedures are outlined next. Finally, a reduction test is completed for each variable in the problem to determine if the value for that variable can be fixed in the optimal solution and therefore removed prior to performing the computationally complex Expanding Core procedure. This full process is demonstrated in Figure 4 and the pseudocode for these procedures is provided in Appendix A.



Initialize ($n_s = 1$)
Calc. U^* and call IMP, REMREPL, REMREPL2, and FEAS to find lower bound (P)
Reduce the problem by fixing all elements of I_0 and I_1 to appropriate values and let $n_s = j$
Initialize I_0 and I_1 as empty and let $j = n_s$
NO
$U^* = P$?
YES
Terminate (Current solution defining P is optimal)
YES
$x_j = 1$ or a break item in $CLU_R(\bar{\lambda})$?
NO
$x_j = 0$ or a break item in $CLU_R(\bar{\lambda})$?
NO
$I_0 \cup I_1 \neq \emptyset$?
NO
Commence Expanding Core Procedure
YES
YES
YES
Solve $CLU_R(\bar{\lambda})$ assuming $x_j = 0$
Solve $CLU_R(\bar{\lambda})$ assuming $x_j = 1$
$I_0 \cap I_1 = \emptyset$ and does fixed items still permit feas.?
NO
Terminate (Current solution defining P is optimal)
YES
$\lfloor CLU_R(\bar{\lambda}) \rfloor_{x_j=0}$ less than or equal to P?
NO
$\lfloor CLU_R(\bar{\lambda}) \rfloor_{x_j=1}$ less than or equal to P?
NO
NO
$|I_0 \cup I_1| \geq n/25$ or $j = n_s$?
YES
YES
Add j to I_1
Add j to I_0
Update j to next item in problem

Figure 4. Flowchart of REDUCE procedure

The first procedure to determine a lower bound on the current DKP instance is titled IMP (Improve) and is designed such that the problem is first made feasible with respect to the knapsack constraint. Items are then added in a greedy manner with the highest priority of first inducing complete feasibility if possible and then with improving the solution quality. Specifically, the best solution vector, $\boldsymbol{x} = (x_1, \ldots, x_n)$, from (4-32) is sorted in a non-descending order according to each variable's efficiency measure. If $\boldsymbol{x}$ is infeasible with respect to the knapsack constraint, variables are greedily removed from the solution starting at the end of the list until knapsack feasibility is obtained. Given this sorted, knapsack feasible vector, excluded variables are added given two cases. If the current vector is not feasible with respect to (4-3), then add that variable regardless of its objective coefficient assuming the solution would still be knapsack feasible. If the current vector is feasible with respect to (4-3), then add that variable only if its objective function coefficient is positive assuming the solution would still be knapsack feasible. The total time complexity of IMP is therefore $O(n)$ excluding sorting. If $\boldsymbol{x}$ now represents a feasible solution to the DKP, update the best lower bound and let its objective value be P.

Given the vector from IMP, the procedure REMREPL (Remove and Replace) is performed which sequentially tests the removal of included items and then greedily fills the remainder of the solution. To outline REMREPL, assume the variables are still sorted according to IMP and let k be the smallest index of the variable that is excluded in the current solution. Define the solution vector $\boldsymbol{x}' = (x_1', \dots, x_n')$ such that $x_i' = 1$ for $i = 1, \dots, k-1$ and $x_i' = 0$ for $i = k, \dots, n$. Starting with $k-1$, remove x_{k-1}' from $\boldsymbol{x}'$ and attempt to fill the vector starting with x_k' according to the same logic given at the end of IMP. Once complete, retain the vector if it is the best *feasible* solution identified thus far. Next, complete the same process by starting with the original vector $\boldsymbol{x}'$ and removing x_{k-2}' and filling the remainder starting with x_k' as detailed in IMP. Continue this removal and filling until x_1' or until a global parameter a items have been tested for removal and subsequent filling. If the best $\boldsymbol{x}'$ is feasible and better than $\boldsymbol{x}$, update the best solution and P. The time complexity of REMREPL is therefore $O(n^2)$.

The procedure REMREPL2 (Remove and Replace 2) is identical to REMREPL except that pairs of items are removed prior to the filling the solution in the manner detailed in IMP. Specifically, REMREPL2 defines the initial solution vector $\boldsymbol{x}'$ the same as REMREPL. Then, all possible pairs of items whose indices are between $\max\{1, k - a'\}$, where a' is another global parameter, are removed from $\boldsymbol{x}'$ prior to filling the remainder of the solution starting with x_k' as described in IMP. In this implementation, $a' = \sqrt{a}$ and therefore REMREPL2 has $O(n^2)$ complexity.

The final procedure, FEAS, is used solely in the case where all of the three aforementioned procedures did not identify a feasible solution with respect to both constraints. Specifically, sort all variables according to their non-increasing values of v_i/w_i with ties broken according to p_i. Initially add sorted variables to the solution vector regardless of the value of p_i until feasibility of both constraints is achieved. After feasibility, continue to add a variable x_i only if $p_i > 0$ and adding the variable does not violate the feasibility of the knapsack constraint. Ignoring the sorting, the time complexity of FEAS is $O(n)$. If FEAS does not find a feasible solution, the entire algorithm can be stopped as the current instance is infeasible. If FEAS does identify a feasible solution, it is likely of low quality based on the construction method so it is recommended that IMP, REMREPL, and REMREPL2 are conducted again to possibly improve the solution.

After these procedures are completed, the optimal solution has been identified and REDUCE (and the DKPSOLVE) is complete if P equals U^*. Otherwise, let I_0 and I_1 store the indices of variables which the procedure has identified as fixed to 0 and 1 respectively for any solution which must exceed P. These are initialized as empty and let

j represent the index of the item currently under investigation. Starting with j, the continuous upper bound $\left\lfloor z\left(C\left(L_R(DKP,\tilde{\lambda})\right)\right)\right\rfloor$ is solved assuming $x_j = 0$ if j is included in the linear solution of the DKP or $x_j = 1$ if j is excluded in the linear solution of the DKP. Let the solution to these problems be denoted as $\left\lfloor z\left(C\left(L_R(DKP,\tilde{\lambda})\right)\right)\right\rfloor_{x_j=0}$ and $\left\lfloor z\left(C\left(L_R(DKP,\tilde{\lambda})\right)\right)\right\rfloor_{x_j=1}$ respectively. Observe that $\tilde{\lambda}$ may not be the optimal multiplier for these problems given the fixed variable, but they should be close enough to the correct multiplier to provide competitive results without the computational burden to correctly determine the true multiplier. If x_j is a break item in the linear solution, both of these values are calculated. If the solution to these problems are less than P, then j should be added to the appropriate storage I_o or I_1. This process continues until the cumulative size of these two lists exceeds $n/25$ or all variables have been investigated.

If the intersection of these lists is non-empty, the process is terminated as the procedure identified that there is no binary value for the break item(s) which will ever result in a solution better than P. Additionally, the process is terminated if fixing the variables according to I_0 and I_1 will result in a problem which cannot be made feasible. Otherwise, if both lists are empty, the problem is fully reduced and EXPCORE is called which is the second phase of the DKPSOLVE procedure. If either list is non-empty, the problem is reduced by fixing all elements of I_0 and I_1 and REDUCE is called with the reduced set of variables (i.e. all those which have yet to be fixed). Observe that REDUCE starts investigating the variables starting where the prior procedure terminated so that all variables are investigated equally.

4.4 DKP expanding core procedure

Once the problem is reduced, the remainder of the problem is solved through a breadth-first Expanding Core procedure, hereafter referred to as EXPCORE, similar to the technique first presented by Pisinger (1997). In EXPCORE, a branch-and-bound tree is investigated in a breadth-first manner such that each branch represents the inclusion/exclusion of items in the reduced DKP solution. It is referred to as the Expanding Core as this branching starts with the break items and then expands to the items which are most likely to be included in the true core of the problem. Both the breadth-first and depth-first Expanding Core approaches were tested. The breadth-first approach performed significantly better and is presented in this section. This differs from the approach used by Martello and Toth (2003) for the two-constraint KP as they used a depth-first branch-and-bound procedure without giving branching priority to the likely core items.

To demonstrate the procedure, assume that the problem has been reduced and there are $n' \leq n$ items for consideration. EXPCORE commences by sorting all n' items according to their non-decreasing efficiency measures as defined in Chapter 3. The values needed to calculate these measures (i.e. the optimal Lagrangian multipliers) are identified during the last calculation of U^* from the Reduction phase. Let b be the index of the first sorted item whose efficiency measure equals one (i.e. one of the possible two break items for the DKP) and let P be inherited from the last REDUCE procedure.

Let $N(s,t)$ represent the set of non-fathomed nodes in the tree where the pair s and t represents the depth of the current tree. Specifically, s and t are the inclusive start and end of the subset of items which have already been branched. For example, $N(b-1, b+1)$ represents the third level of the tree which has at most eight elements if no nodes have been fathomed in prior levels of the tree. For any $j \in N(s,t)$, the node represents the DKP solution such that $x_i = 1$ for all $i \in [1, s-1]$, $x_i = 0$ for all $i \in [t+1, n']$, and x_i for all $i \in [s,t]$ are set based on the path in the tree leading to node j. Let the vector be noted as $\boldsymbol{x}_j = \{x_1^j, \ldots, x_{n'}^j\}$ and let P^j, W^j, and V^j represent the objective value, knapsack constraint left hand side, and demand constraint right hand side corresponding with vector $\boldsymbol{x}_j$ respectively. Additionally, let U^j represent the upper bound on the optimal binary solution of node j assuming that x_i for all $i \in [s,t]$ are fixed to their branched values for the node. The methodology to calculate U^j is presented shortly and let $\tilde{\lambda}^j$ and $\tilde{\mu}^j$ represent the multipliers used to calculate U^j.

To assist in fathoming nodes in the branch-and-bound tree, $N(s,t)$ is kept ordered such that dominated nodes can be easily identified. A node $j \in N(s,t)$ is dominated by $k \in N(s,t)$ if $P^k \geq P^j$, $W^k \leq W^j$, and $V^k \geq V^j$ as the future branches leading from j can be shown to never provide a better solution than k or one of its future branches. Hence, j is listed before k in $N(s,t)$ if $P^j > P^k$ or $P^j = P^k$ and $W^j < W^k$ or $P^j = P^k$ and $W^j = W^k$ and $V^j > V^k$. A dominance test for a node $j \in N(s,t)$ therefore requires comparing j to only the preceding nodes in $N(s,t)$.

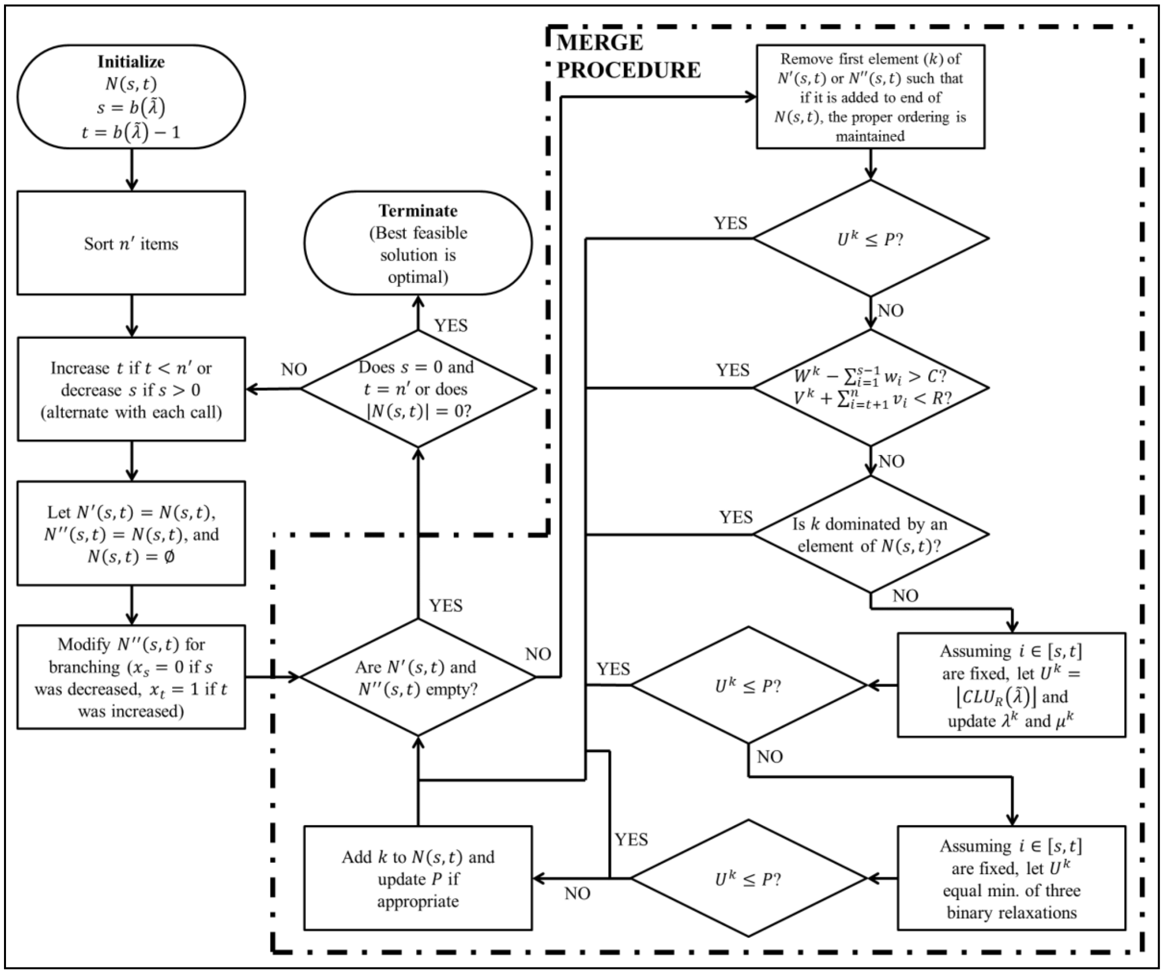
To evaluate each level of the Expanding Core branch-and-bound tree, a procedure MERGE is performed which sorts, analyzes, and fathoms applicable nodes in the current level of the tree. On input, MERGE receives the current list of *possible* nodes for that level in the tree partitioned into two sets $N'(s,t)$ and $N''(s,t)$. Let $N'(s,t)$ equal the final set of non-fathomed nodes from the prior level of the tree. Hence, $N'(s,t)$ represents branching on $x_s = 1$ if the prior set of nodes is $N(s+1,t)$ or $N'(s,t)$ represents branching on $x_t = 0$ if the prior set of nodes is $N(s,t-1)$. Let $N''(s,t)$ represent the set of nodes for the opposite branches. $N''(s,t)$ is therefore the same as the parent set of nodes with only minor modifications to $\boldsymbol{x}_j$, P^j, W^j, and V^j. In addition, let U^j, $\tilde{\lambda}^j$, and $\tilde{\mu}^j$ be inherited from the parent node. Observe that if the list of parent of nodes is correctly sorted, $N'(s,t)$ and $N''(s,t)$ will each be sorted correctly.

Given these lists, a node j is selected from the start of $N'(s,t)$ or $N''(s,t)$ and is evaluated for addition to $N(s,t)$ through five fathoming tests. Specifically, the node is selected from either list such that $N(s,t)$ is ordered correctly if the node is not fathomed and added to the current end of $N(s,t)$. Since $N'(s,t)$ and $N''(s,t)$ are sorted correctly by construction, this only requires comparing the first non-analyzed entries in either list. Once $j \in N'(s,t) \cup N''(s,t)$ is determined, the first fathoming test is if $U^j \leq P$. At this phase, U^j is inherited from the parent node which was unfathomed, but P may have increased since the parent node was tested. The second fathoming test validates that j or one of its eventual children can be feasible. Specifically, j is fathomed if $W^j - \sum_{i=1}^{s-1} w_i > C$ or $V^j + \sum_{i=t+1}^{n'} v_i < R$. The third fathoming test is to test whether j is dominated by any of the non-fathomed nodes already added to $N(s,t)$. If a dominating node is identified, then j is fathomed.

If j is not yet fathomed, the remaining two tests update the inherited U^j. For each of these tests, it is assumed that x_i for all $i \in [s,t]$ are fixed according to the node's path in the tree. The fourth fathoming test is to let $U^j = \lfloor CLU_R(\tilde{\lambda}) \rfloor$ using CDKPSOLVE and to fathom the node if $U^j \leq P$. This procedure is necessary as it updates the inherited λ^j and μ^j which is needed for the next test. Finally, the last fathoming test is to let $U^j = \min\left(\left\lfloor z\left(L_R(DKP, \lambda^j)\right)\right\rfloor, \left\lfloor z\left(L_C(DKP, \mu^j)\right)\right\rfloor, z\left(S(DKP, \lambda^j, \mu^j)\right)\right)$ still assuming that x_i for all $i \in [s, t]$ are fixed accordingly. The node is again fathomed if $U^j \leq P$.

If the node passes all five fathoming tests, two operations are completed. First, the node is added to the end of $N(s, t)$ to maintain the proper ordering of the list. Second, the value of P is updated if the solution vector represented by the node (i.e. $x_i = 1$ for all $i < s$, $x_i = 0$ for all $i > t$, and x_i is fixed for all $i \in [s, t]$ based on the node's path) is feasible and $P^j > P$ or if the solution vector associated with U^j is feasible and exceeds P. In such a situation, the new solution vector is tested for improvement by FEAS, REMREPL, and REMREPL2 to see if the lower bound can be improved further.

The MERGE procedure terminates once all nodes in $N'(s, t)$ and $N''(s, t)$ have been analyzed, fathomed, and added to $N(s, t)$. If MERGE terminates and $N(s, t) = \emptyset$ or $s = 1$ and $t = n'$, then EXPCORE is complete as all possible paths within the tree have been fathomed or have reached their final leaf nodes. A graphical representation of EXPCORE and MERGE is shown in Figure 5 and the pseudocode for the procedure is provided in Appendix A along with the pseudocode for the procedures outlined in REDUCE.



MERGE PROCEDURE
Initialize
$N(s,t)$
$s = b(\tilde{\lambda})$
$t = b(\tilde{\lambda}) - 1$
Sort n' items
Increase t if $t < n'$ or decrease s if $s > 0$ (alternate with each call)
Let $N'(s,t) = N(s,t)$, $N''(s,t) = N(s,t)$, and $N(s,t) = \emptyset$
Modify $N''(s,t)$ for branching ($x_s = 0$ if s was decreased, $x_t = 1$ if t was increased)
Terminate (Best feasible solution is optimal)
Does $s = 0$ and $t = n'$ or does $|N(s,t)| = 0$?
Are $N'(s,t)$ and $N''(s,t)$ empty?
Add k to $N(s,t)$ and update P if appropriate
Remove first element (k) of $N'(s,t)$ or $N''(s,t)$ such that if it is added to end of $N(s,t)$, the proper ordering is maintained
$U^k \leq P$?
$W^k - \sum_{i=1}^{s-1} w_i > C$? $V^k + \sum_{i=t+1}^{n} v_i < R$?
Is k dominated by an element of $N(s,t)$?
Assuming $i \in [s,t]$ are fixed, let $U^k = \lfloor CLU_R(\tilde{\lambda}) \rfloor$ and update λ^k and μ^k
Assuming $i \in [s,t]$ are fixed, let U^k equal min. of three binary relaxations
YES
NO

Figure 5. Flowchart of EXPCORE procedure

4.5 DKPSOLVE algorithmic improvements

While the presented DKPSOLVE procedure fully solves a DKP to optimality, several algorithmic improvements were added to improve the computational performance. These improvements are described in the next two subsections.

4.5.1 CDKPSOLVE improvements

When performing CDKPSOLVE, the proof in Theorem 2 calculated the median value in the remaining set Λ using the method from Blum et al. (1973). However, this method proved unsatisfactory with respect to overall solution time. Instead, each $\lambda \in \Lambda$ was assigned to a storage bin based on its value rounded down to the nearest hundredth. Storage was allocated for rounded values from 0.00 to 100.00 and any $\lambda \in \Lambda$ greater than 100 was assigned to the last bin. Since fully sorting all elements of Λ into these bins would be inefficient, a doubly linked list was used to identify all of the elements of Λ in each bin. To best utilize this storage, (d) in CDKPSOLVE was replaced such that elements of Λ were tested until the true value of $\tilde{\lambda}$ was known within a maximum range of ± 0.5 (i.e. increasing/decreasing tested elements of Λ by 1.00 during each call). Once this range was determined, elements of Λ were tested until the true value of $\tilde{\lambda}$ was known within a maximum range of ± 0.01. This was accomplished by testing values at the midpoint of the current range until only two bins remained which were known to hold the true value of $\tilde{\lambda}$. While computational complexity is difficult to determine for this process, this new procedure significantly improved performance in preliminary tests.

To best track the maximum range on the true value of $\tilde{\lambda}$ using this bin strategy, CDKPSOLVE was modified such that step (g) no longer removed elements from Λ. Instead, the maximum and minimum possible value for $\tilde{\lambda}$ were tracked based on the returned values of $LHS_{MIN}(\lambda)$ and $LHS_{MAX}(\lambda)$. These limits were initialized as 0 and ∞ at the start of CDKPSOLVE. Furthermore, once the two bins were identified which held the true value for $\tilde{\lambda}$, the median of the first three values in the bins which were between the current limits were tested next within CDKPSOLVE. Such an approach is similar to the technique used by Balas and Zemel (1980).

After implementing this approach, preliminary computational tests identified that the repeated searches through Λ in CDKPSOLVE comprised a majority of the solution time for simpler test cases. To avoid such computational burden, all elements of Λ which were based on a variable i were removed whenever i was permanently discarded or fixed within the procedure. This occurred when an item was selected for reduction during REDUCE and when an item served as the branching criteria in EXPCORE. To ensure this removal was efficient, the location of each element in Λ based on each decision variable was tracked using doubly linked lists.

To further improve the performance of CDKPSOLVE, elements of Λ were calculated and added as needed. Specifically, if all possible values within both aforementioned bins were tested and the optimal value had yet to be identified (only possible when all elements of Λ have not be added), then more values were added according to Lemma 2. To implement this procedure, Λ was initialized with 0 and all elements of (b) in Lemma 2. To ensure that the most likely multipliers were added to Λ as early as possible, the decision variables were reordered such that the pair of elements for the optimal calculation of (c) in Lemma 2 were at the beginning of this ordering prior to the first call of CDKPSOLVE. This is equivalent to identifying the two break items from the optimal CDKPSOLVE solution as these variables can be shown to provide the necessary (c) in Lemma 2.

To identify these variables, the optimal $\tilde{\lambda}$ and $\tilde{\mu}$ were estimated by alternatingly solving $LU_R(\lambda)$ and $LU_C(\mu)$ where the ratio of the objective coefficient over the knapsack constraint coefficient of the break item from $LU_R(\lambda)$ was used as the next multiplier when solving $LU_C(\mu)$ and the ratio of the objective coefficient over the demand constraint coefficient of the break item from $LU_C(\mu)$ was used as the next multiplier when solving $LU_R(\lambda)$. This alternating process was started by solving $LU_R(0)$. The estimates obtained during this process will approach the optimal $\tilde{\lambda}$ and $\tilde{\mu}$. Within this implementation, this alternating procedure was stopped once the pair of break items identified in the current solution of $LU_R(\lambda)$ and $LU_C(\mu)$ equaled the same break items in the prior pair of solutions. With these estimates of $\tilde{\lambda}$ and $\tilde{\mu}$, the efficiency measures for each variable were estimated and were sorted such that those which were closest to 1 were listed first. Hence, if $\tilde{\lambda}$ and $\tilde{\mu}$ were estimated exactly, the break items will be the first two items in this list and the first element added to Λ will be the optimal multiplier needed for the first call to CDKPSOLVE. This reordering only occurs once prior to the first call of REDUCE.

The final improvement to the CDKPSOLVE procedure was in regards to step (b) which recommends solving $LU_R(\lambda)$ by initializing $\lambda = 0$. However, it was more efficient to initialize λ to the last known optimal multiplier if one was available. During the Reduction phase, this was equal to the optimal $\tilde{\lambda}$ identified from the prior call to CDKPSOLVE. During the Expanding Core phase, the optimal multiplier λ^j for a node j is specific to the node. Hence, CDKPSOLVE initialized λ equal to the optimal multiplier of the parent node. This tends to be efficient as a child node will often share the same or similar multiplier value as the parent node.

4.5.2 EXPCORE improvements

The only Expanding Core procedure improvement made was to avoid the fathoming tests which require duplicate calculations. For instance, assume that for any node in the branch-and-bound tree, the fourth fathoming test, solving CDKPSOLVE with items s through t fixed, and the fifth fathoming test, calculating all binary Lagrangian and surrogate relaxations, identified that item $x_{s-1} = 1$ in all four solutions and the node is unfathomed. Assuming that item $s - 1$ is branched in the next level of the tree, completing the fourth and fifth fathoming tests for the child node in $N'(s, t)$ of this parent node will result in the same solutions. To avoid these duplicate calculations, each time an unfathomed node passes the fourth and fifth fathoming tests, each of the four solutions are analyzed and the largest index $s' < s$ for which $x_{s'} \neq 1$ in any of the four solutions is recorded as well as the smallest index $t' > t$ for which $x_{t'} \neq 0$. These values are then inherited by the offspring nodes in $N'(s, t)$ such that the fourth and fifth fathoming tests were only completed if s' or t' were the most recent branched items in the tree.

4.6 DKPSOLVE computational tests

DKPSOLVE was tested using randomly generated test cases since benchmark cases do not exist for the DKP. The algorithm was implemented in C++ and run on a 2.2 GHz processor with 8 GB of RAM. All test cases were also solved using commercial software, specifically IBM CPLEX version 12.6, as a comparison solution method. In total, seven different data Cases were generated for testing. These Cases were developed by applying common methodologies for generating test instances as seen in the two-constraint KP (Martello and Toth 2003) and binary KPs (Balas and Zemel 1980; Pisinger 1997; Martello, Pisinger, and Toth 1999). The seven data Cases were designed to test different levels of correlation between the constraint and objective coefficients:

A: p_i, w_i, and v_i are integer uniform random in $[1, U]$,

B: w_i, and v_i are integer uniform random in $[1, U]$ and p_i is integer uniform random in $[w_i - v_i - U/5\,, w_i - v_i + U/5]$

C: w_i, and v_i are integer uniform random in $[1, U]$ and p_i is integer uniform random in $[w_i - v_i - U/20\,, w_i - v_i + U/20]$

D: w_i, and p_i are integer uniform random in $[1, U]$ and v_i is integer uniform random in $[U - w_i, U - w_i + U/5]$

E: w_i, and p_i are integer uniform random in $[1, U]$ and v_i is integer uniform random in $[U - w_i, U - w_i + U/20]$

F: w_i, is integer uniform random in $[1, U]$, v_i is integer uniform random in $[U - w_i, U - w_i + U/5]$, and p_i is integer uniform in $[w_i - v_i - U/5\,, w_i - v_i + U/5]$

G: w_i, is integer uniform random in $[1, U]$, v_i is integer uniform random in $[U - w_i, U - w_i + U/20]$, and p_i is integer uniform in $[w_i - v_i - U/20\,, w_i - v_i + U/20]$.

Hence, Case A feature no coefficient correlations, Cases B and C have correlation between one constraint and the objective, Cases D and E have correlation between the constraints only, and Cases F and G have correlation between all coefficients.

For these Cases, seven values of n were tested (100, 500, 1000, 5000, 10000, 50000, and 100000) and two values for U were tested (100 and 1000). For each of these combinations, the constraint limits were calculated as $C = M \sum_{i=1}^{n} w_i$ and $R = 0.5 \sum_{i=1}^{n} v_i$. For all test combinations, three possible values of M were tested (0.25, 0.45, and 0.65). The combinations of Case, n, U, and M are hereafter referred to as the test combinations. For each test combination, 10 data instances were generated. Each instance had a time limit of 600 seconds for both solution methods. The only other parameter is a which was set to 1000 for all test combinations as preliminary testing identified this setting provided satisfactory results.

The results from these tests are shown in Table 9 through Table 13. For the first three of these tables, any entries in bold represent test combinations in which DKPSOLVE outperformed CPLEX for the indicated measure. Table 9 provides a pairwise comparison between CPLEX and DKPSOLVE by counting the number of instances in which DKPSOLVE terminated equal to or faster than CPLEX within the timing tolerances. If all of the ten test instances were not solved within the time limit by either solution technique, the quantity of instances whose optimal solution is guaranteed is indicated in parentheses. Table 10 and Table 11 show the average and median solution time for each test combination solved by DKPSOLVE respectively. The same measures for CPLEX are shown in both tables in parentheses. Note that the instances which terminated due to the time limit are also included in the measures for Table 10 and Table 11. Finally, Table 12 provides the average absolute difference between the solution at the end of the Reduction phase (i.e. P) and the optimal solution while Table 13 provides the average percentage of the total time DKPSOLVE required for the Reduction phase for each test combination. If the optimal solution is not guaranteed (as noted by the count of instances in Table 9 shown in parentheses), the calculations in Table 12 use the best known solution at termination. This is likely the optimal or near optimal solution, but it cannot be guaranteed.

Table 9. The count of test instances DKPSOLVE terminated simultaneously or prior to CPLEX.

$Case, U, M \backslash n$	100	500	1000	5000	10000	50000	100000
(A, 100, 0.25)	**10**	**10**	**10**	**10**	**10**	**6**	**8**
(B, 100, 0.25)	**10**	**10**	**10**	**10**	**10**	**5**	**6**
(C, 100, 0.25)	**10**	**10**	**10**	**6**	**6**	**10**	**9**
(D, 100, 0.25)	**8**	**9**	**9**	**8**	**8**	3	2
(E, 100, 0.25)	**9**	**10**	**10**	**3 (6)**	3 (7)	2 (7)	0 (9)
(F, 100, 0.25)	**6**	**9**	**6**	**8**	**8**	3	0
(G, 100, 0.25)	**10**	**3 (3)**	**2 (3)**	0 (8)	0 (8)	1	0
(A, 100, 0.45)	**10**	**10**	**10**	**9**	**10**	**10**	**10**
(B, 100, 0.45)	**10**	**10**	**10**	**9**	**7**	**6**	**7**
(C, 100, 0.45)	**9**	**10**	**9**	**7**	**8**	**9**	**10**
(D, 100, 0.45)	**10**	**10**	**9**	**10**	**6**	4	**8**
(E, 100, 0.45)	**10**	**10**	**9**	**9**	**9**	**6**	2 (7)
(F, 100, 0.45)	**8**	**8**	**9**	4	**7**	2	1
(G, 100, 0.45)	**4 (4)**	4 (9)	1 (3)	1 (8)	0 (9)	0	0
(A, 100, 0.65)	**10**	**10**	**10**	**10**	**10**	**10**	**10**
(B, 100, 0.65)	**10**	**10**	**10**	**10**	**7**	**7**	**8**
(C, 100, 0.65)	**10**	**10**	**10**	**6**	**8**	**7**	**7**
(D, 100, 0.65)	**10**	**10**	**10**	**10**	**10**	**10**	**10**
(E, 100, 0.65)	**10**	**10**	**10**	**8**	**9**	**10**	**10**
(F, 100, 0.65)	**9**	**9**	**6**	4	2	2	3
(G, 100, 0.65)	**8 (8)**	**6 (7)**	**5 (6)**	**3 (6)**	2 (8)	0	0
(A, 1000, 0.25)	**10**	**10**	**10**	**10**	**10**	**10**	2
(B, 1000, 0.25)	**9**	**10**	**10**	**10**	**8**	**9**	**5**
(C, 1000, 0.25)	**10**	**10**	**9**	**9**	**10**	**9**	**5**
(D, 1000, 0.25)	**9**	4	**6**	**9 (9)**	**7 (7)**	4 (9)	3 (9)
(E, 1000, 0.25)	**10**	**10**	**6 (7)**	**3 (3)**	**2 (2)**	**2 (3)**	0 (3)
(F, 1000, 0.25)	**10**	**8**	**10**	**8 (9)**	**10**	**7 (8)**	**6 (9)**
(G, 1000, 0.25)	**1 (2)**	**2 (2)**	**4 (4)**	**3 (4)**	**2 (3)**	**2 (4)**	0 (4)
(A, 1000, 0.45)	**10**	**9**	**9**	**9**	**10**	2	0
(B, 1000, 0.45)	**9**	**9**	**10**	**10**	**9**	**9**	**6**
(C, 1000, 0.45)	**9**	**10**	**10**	**10**	**10**	**9**	**7**
(D, 1000, 0.45)	**10**	**8**	**9**	**5**	**7**	2	0
(E, 1000, 0.45)	**9**	**8**	**7**	**6 (7)**	**2 (2)**	0 (4)	1 (4)
(F, 1000, 0.45)	**6**	**9**	**8 (9)**	**10**	**9 (9)**	**5 (7)**	4 (9)
(G, 1000, 0.45)	**2 (2)**	**2 (3)**	**1 (1)**	**5 (5)**	**2 (2)**	**2 (2)**	1 (5)
(A, 1000, 0.65)	**10**	**10**	**10**	**10**	**10**	**10**	**10**
(B, 1000, 0.65)	**10**	**10**	**10**	**10**	**10**	**9**	**5**
(C, 1000, 0.65)	**8**	**9**	**10**	**10**	**8**	**8**	**8**
(D, 1000, 0.65)	**10**	**10**	**10**	**10**	**10**	**10**	**10**
(E, 1000, 0.65)	**10**	**10**	**10**	**10**	**10**	**10**	**10**
(F, 1000, 0.65)	**6**	**9**	**9 (9)**	**9 (9)**	**8 (9)**	**5 (7)**	2 (6)
(G, 1000, 0.65)	**5 (5)**	**4 (4)**	**1 (1)**	**4 (4)**	**3 (4)**	0 (0)	0 (0)

Table 10. The ratio of the average algorithmic termination time for CPLEX over the average algorithm termination time for DKPSOLVE.

$Case, U, M \backslash n$	100	500	1000	5000	10000	50000	100000
(A, 100, 0.25)	**12.71**	**5.21**	**4.95**	**4.13**	**3.00**	0.31	0.50
(B, 100, 0.25)	**3.81**	**6.45**	**4.78**	**4.50**	**4.05**	0.47	0.13
(C, 100, 0.25)	**5.60**	**10.63**	**5.66**	0.68	0.02	**12.01**	0.72
(D, 100, 0.25)	**1.33**	**6.70**	**1.39**	0.76	**1.18**	0.39	0.10
(E, 100, 0.25)	**1.45**	**3.53**	**5.15**	**1.10**	**1.22**	0.78	0.29
(F, 100, 0.25)	0.76	**2.45**	0.28	**1.61**	0.03	0.03	0.02
(G, 100, 0.25)	**1.83**	**1.07**	**1.12**	0.41	0.65	0.01	0.01
(A, 100, 0.45)	**11.12**	**10.62**	**4.88**	**2.04**	**2.85**	**32.68**	**13.55**
(B, 100, 0.45)	**3.18**	**4.76**	**7.78**	**1.70**	0.44	0.05	0.10
(C, 100, 0.45)	**2.50**	**5.52**	**4.29**	0.31	0.82	**3.55**	**17.00**
(D, 100, 0.45)	**10.69**	**2.01**	**1.28**	**6.77**	0.71	0.40	0.52
(E, 100, 0.45)	**17.39**	**2.18**	**17.66**	**43.22**	**6.20**	**3.89**	0.93
(F, 100, 0.45)	**1.22**	0.97	**1.47**	0.09	0.31	0.02	0.02
(G, 100, 0.45)	**1.25**	0.94	0.89	0.46	0.15	0.01	0.00
(A, 100, 0.65)	**806.00**	**206.50**	**303.33**	**64.90**	**66.76**	**67.85**	**94.37**
(B, 100, 0.65)	**3.23**	**6.75**	**5.63**	**2.57**	**1.00**	0.25	0.09
(C, 100, 0.65)	**3.90**	**7.73**	**4.67**	0.20	0.21	0.03	0.05
(D, 100, 0.65)	**708.00**	**150.80**	**110.38**	**84.81**	**68.56**	**70.70**	**89.82**
(E, 100, 0.65)	**229.67**	**789.00**	**217.25**	**1.57**	0.57	**79.50**	**73.91**
(F, 100, 0.65)	**1.25**	**1.75**	0.30	0.42	0.11	0.01	0.02
(G, 100, 0.65)	**1.50**	**1.38**	**1.00**	0.75	0.62	0.01	0.00
(A, 1000, 0.25)	**6.49**	**3.37**	**2.69**	**3.53**	**2.98**	**2.38**	0.94
(B, 1000, 0.25)	**1.60**	**3.34**	**3.90**	**9.37**	**3.40**	**2.87**	**1.22**
(C, 1000, 0.25)	**2.19**	**3.99**	**4.12**	**8.04**	**11.54**	**2.87**	0.90
(D, 1000, 0.25)	0.85	0.68	**1.89**	**1.09**	**1.08**	**1.02**	**1.50**
(E, 1000, 0.25)	**2.78**	**9.94**	**1.59**	**1.00**	**1.01**	**1.12**	0.87
(F, 1000, 0.25)	**1.59**	**1.36**	**2.50**	**1.18**	**5.92**	**1.60**	**1.11**
(G, 1000, 0.25)	**1.07**	**1.02**	**1.02**	**1.06**	**1.02**	0.96	0.87
(A, 1000, 0.45)	**17.78**	**2.71**	**2.03**	**1.85**	**2.07**	0.64	0.24
(B, 1000, 0.45)	**2.00**	**2.55**	**3.27**	**6.94**	**5.40**	**3.38**	**1.46**
(C, 1000, 0.45)	**2.67**	**5.60**	**4.51**	**12.55**	**10.38**	**2.75**	**1.83**
(D, 1000, 0.45)	**12.09**	**1.35**	**2.26**	0.99	0.61	0.88	0.18
(E, 1000, 0.45)	**3.93**	**1.06**	**2.36**	**1.32**	**1.04**	0.99	0.97
(F, 1000, 0.45)	**1.59**	**1.48**	**1.23**	**7.88**	**1.82**	**1.05**	0.78
(G, 1000, 0.45)	**1.06**	0.96	**1.00**	**1.25**	**1.00**	**1.10**	0.86
(A, 1000, 0.65)	**843.00**	**762.00**	**122.38**	**46.38**	**21.14**	**7.67**	**9.54**
(B, 1000, 0.65)	**2.20**	**2.52**	**4.17**	**8.09**	**4.62**	**2.70**	0.94
(C, 1000, 0.65)	**2.72**	**3.49**	**5.37**	**7.48**	**11.04**	**4.59**	**1.71**
(D, 1000, 0.65)	**813.00**	**207.50**	**112.50**	**45.71**	**43.94**	**10.11**	**8.14**
(E, 1000, 0.65)	**183.25**	**147.00**	**139.40**	**49.34**	**20.15**	**24.57**	**7.75**
(F, 1000, 0.65)	0.77	**1.05**	**1.27**	**1.62**	**1.10**	**1.12**	0.88
(G, 1000, 0.65)	**1.13**	**1.02**	**1.00**	**1.19**	0.95	**1.00**	**1.00**

Table 11. The ratio of the median algorithmic termination time for CPLEX over the median algorithm termination time for DKPSOLVE.

$Case, U, M \backslash n$	100	500	1000	5000	10000	50000	100000
(A, 100, 0.25)	**13.94**	**5.96**	**4.85**	**3.70**	**2.68**	**2.35**	**19.35**
(B, 100, 0.25)	**5.53**	**9.04**	**5.83**	**4.44**	**2.92**	**1.22**	**32.75**
(C, 100, 0.25)	**8.41**	**11.53**	**8.53**	**7.75**	**2.02**	**28.67**	**18.04**
(D, 100, 0.25)	**2.91**	**3.66**	**1.71**	**2.39**	**1.31**	0.30	0.16
(E, 100, 0.25)	**2.28**	**2.22**	**2.58**	**1.78**	0.91	0.09	0.01
(F, 100, 0.25)	**1.00**	**1.00**	**2.32**	**2.87**	**1.20**	0.04	0.01
(G, 100, 0.25)	**3.89**	**1.00**	**1.00**	0.09	0.02	0.01	0.00
(A, 100, 0.45)	**6.13**	**9.03**	**5.23**	**2.48**	**2.77**	**35.97**	**35.13**
(B, 100, 0.45)	**4.15**	**6.93**	**8.32**	**3.31**	**1.43**	**7.89**	**34.59**
(C, 100, 0.45)	**3.08**	**6.34**	**6.80**	**7.12**	**27.63**	**35.98**	**35.95**
(D, 100, 0.45)	**10.50**	**6.39**	**3.91**	**2.11**	**1.93**	0.35	**25.45**
(E, 100, 0.45)	**78.50**	**7.57**	**1.36**	0.56	**37.69**	0.32	0.27
(F, 100, 0.45)	**1.09**	**9.42**	**4.54**	0.22	0.98	0.05	0.01
(G, 100, 0.45)	**1.00**	0.63	**1.00**	0.01	0.01	0.00	0.00
(A, 100, 0.65)	**8.15**	**8.15**	**8.90**	**127.00**	**87.20**	**73.38**	**96.15**
(B, 100, 0.65)	**4.33**	**11.91**	**6.18**	**3.04**	**1.64**	**5.90**	**32.98**
(C, 100, 0.65)	**5.54**	**11.25**	**7.66**	**3.86**	**8.50**	**2.37**	**40.50**
(D, 100, 0.65)	**6.95**	**147.00**	**87.50**	**135.00**	**73.60**	**83.82**	**89.36**
(E, 100, 0.65)	**6.85**	**4.60**	**8.40**	**30.14**	**71.67**	**83.64**	**73.22**
(F, 100, 0.65)	**8.61**	**4.05**	**3.83**	0.55	0.46	0.01	0.01
(G, 100, 0.65)	**3.16**	**4.55**	0.22	0.62	0.20	0.01	0.00
(A, 1000, 0.25)	**6.47**	**3.99**	**2.32**	**4.84**	**3.67**	**1.99**	0.68
(B, 1000, 0.25)	**2.15**	**3.16**	**4.50**	**15.57**	**5.10**	**2.61**	**1.61**
(C, 1000, 0.25)	**3.87**	**4.88**	**4.86**	**8.15**	**12.34**	**4.48**	**1.03**
(D, 1000, 0.25)	**8.49**	0.87	0.79	**3.2**	**1.32**	0.98	0.51
(E, 1000, 0.25)	**3.88**	**11.61**	**1.80**	**1.00**	**1.00**	**1.00**	**1.00**
(F, 1000, 0.25)	**2.47**	**5.42**	**2.23**	**5.33**	**24.39**	**4.85**	**1.36**
(G, 1000, 0.25)	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
(A, 1000, 0.45)	**98.33**	**5.75**	**2.56**	**1.92**	**2.48**	0.54	0.26
(B, 1000, 0.45)	**2.26**	**4.01**	**3.36**	**10.52**	**6.75**	**3.43**	**1.71**
(C, 1000, 0.45)	**2.24**	**4.50**	**8.65**	**19.99**	**9.70**	**2.79**	**1.62**
(D, 1000, 0.45)	**7.90**	**8.50**	**4.22**	0.80	**1.73**	0.41	0.19
(E, 1000, 0.45)	**12.36**	**3.82**	0.95	**1.56**	**1.00**	**1.00**	**1.00**
(F, 1000, 0.45)	0.48	**5.65**	**1.55**	**7.18**	**3.84**	**1.01**	**1.80**
(G, 1000, 0.45)	**1.00**	**1.00**	**1.00**	**1.54**	**1.00**	**1.00**	0.95
(A, 1000, 0.65)	**8.20**	**7.60**	**89.50**	**34.88**	**25.21**	**9.08**	**8.04**
(B, 1000, 0.65)	**1.97**	**4.33**	**6.16**	**11.45**	**5.21**	**2.80**	0.83
(C, 1000, 0.65)	**3.12**	**4.51**	**12.40**	**5.08**	**12.00**	**4.62**	**1.36**
(D, 1000, 0.65)	**7.30**	**7.65**	**95.50**	**43.00**	**57.40**	**12.90**	**9.51**
(E, 1000, 0.65)	**7.25**	**5.85**	**140.00**	**38.50**	**19.06**	**31.72**	**10.00**
(F, 1000, 0.65)	**1.25**	**2.15**	**1.65**	**7.58**	**20.66**	**2.47**	0.65
(G, 1000, 0.65)	**1.21**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

Table 12. The average absolute difference between the known optimal/best solution and the solution upon termination of the Reduction phase.

$Case, U, M^{\backslash n}$	100	500	1000	5000	10000	50000	100000
(A, 100, 0.25)	6	4	5	1	1	0	0
(B, 100, 0.25)	7	7	6	2	2	1	1
(C, 100, 0.25)	8	4	1	1	1	0	2
(D, 100, 0.25)	4	4	8	5	5	5	3
(E, 100, 0.25)	2	3	1	3	3	6	7
(F, 100, 0.25)	33	23	18	17	17	18	18
(G, 100, 0.25)	13	24	26	24	24	33	55
(A, 100, 0.45)	2	1	1	1	0	0	0
(B, 100, 0.45)	10	6	6	1	2	0	0
(C, 100, 0.45)	4	4	2	1	0	0	0
(D, 100, 0.45)	1	1	1	1	1	0	0
(E, 100, 0.45)	2	0	0	0	0	0	1
(F, 100, 0.45)	22	19	13	16	14	14	25
(G, 100, 0.45)	12	17	29	19	24	40	32
(A, 100, 0.65)	0	0	0	0	0	0	0
(B, 100, 0.65)	6	5	4	1	0	0	3
(C, 100, 0.65)	6	3	3	0	0	9	3
(D, 100, 0.65)	0	0	0	0	0	0	0
(E, 100, 0.65)	0	0	0	0	0	0	0
(F, 100, 0.65)	18	18	18	15	10	9	15
(G, 100, 0.65)	7	15	17	28	46	83	31
(A, 1000, 0.25)	101	47	69	37	28	21	11
(B, 1000, 0.25)	126	78	49	24	22	16	12
(C, 1000, 0.25)	86	44	32	25	15	12	7
(D, 1000, 0.25)	47	90	89	56	80	53	47
(E, 1000, 0.25)	48	20	26	44	61	54	64
(F, 1000, 0.25)	201	147	299	74	73	167	137
(G, 1000, 0.25)	142	196	201	217	214	151	152
(A, 1000, 0.45)	25	19	16	10	9	5	4
(B, 1000, 0.45)	139	66	51	31	21	12	14
(C, 1000, 0.45)	63	38	32	18	17	9	7
(D, 1000, 0.45)	7	5	9	17	8	16	8
(E, 1000, 0.45)	19	3	6	7	6	8	6
(F, 1000, 0.45)	185	248	209	131	121	211	105
(G, 1000, 0.45)	102	198	223	150	262	200	144
(A, 1000, 0.65)	0	0	0	0	0	0	0
(B, 1000, 0.65)	151	57	52	29	22	14	14
(C, 1000, 0.65)	68	50	30	19	16	10	6
(D, 1000, 0.65)	0	0	0	0	0	0	0
(E, 1000, 0.65)	0	0	0	0	0	0	0
(F, 1000, 0.65)	146	137	204	156	92	118	167
(G, 1000, 0.65)	79	167	273	204	162	203	197

Table 13. The average time ratio of the Reduction phase of DKPSOLVE over the complete DKPSOLVE time.

$Case, U, M^{\backslash n}$	100	500	1000	5000	10000	50000	100000
(A, 100, 0.25)	0.28	0.29	0.35	0.83	0.85	0.25	0.36
(B, 100, 0.25)	0.09	0.29	0.29	0.63	0.85	0.39	0.15
(C, 100, 0.25)	0.08	0.53	0.37	0.08	0.01	1.00	0.59
(D, 100, 0.25)	0.05	0.00	0.01	0.04	0.51	0.26	0.30
(E, 100, 0.25)	0.06	0.01	0.00	0.00	0.00	0.02	0.09
(F, 100, 0.25)	0.00	0.00	0.00	0.19	0.01	0.04	0.09
(G, 100, 0.25)	0.00	0.00	0.00	0.00	0.00	0.03	0.12
(A, 100, 0.45)	0.54	0.85	0.70	0.98	0.99	1.00	1.00
(B, 100, 0.45)	0.05	0.20	0.39	0.31	0.13	0.04	0.13
(C, 100, 0.45)	0.03	0.19	0.25	0.04	0.15	0.88	1.00
(D, 100, 0.45)	0.57	0.12	0.06	0.17	0.33	0.99	0.97
(E, 100, 0.45)	0.65	0.13	0.02	0.15	0.01	0.11	0.13
(F, 100, 0.45)	0.00	0.00	0.00	0.00	0.02	0.06	0.12
(G, 100, 0.45)	0.00	0.00	0.00	0.00	0.00	0.02	0.10
(A, 100, 0.65)	1.00	1.00	1.00	1.00	1.00	1.00	0.99
(B, 100, 0.65)	0.09	0.26	0.41	0.63	0.28	0.21	0.10
(C, 100, 0.65)	0.04	0.33	0.27	0.03	0.04	0.03	0.09
(D, 100, 0.65)	1.00	1.00	1.00	1.00	1.00	1.00	0.99
(E, 100, 0.65)	1.00	1.00	1.00	1.00	1.00	1.00	0.99
(F, 100, 0.65)	0.00	0.00	0.00	0.03	0.02	0.04	0.14
(G, 100, 0.65)	0.00	0.00	0.00	0.00	0.00	0.04	0.12
(A, 1000, 0.25)	0.06	0.15	0.08	0.08	0.13	0.77	0.92
(B, 1000, 0.25)	0.03	0.03	0.02	0.10	0.10	0.56	0.61
(C, 1000, 0.25)	0.01	0.01	0.01	0.06	0.17	0.43	0.27
(D, 1000, 0.25)	0.02	0.00	0.00	0.00	0.00	0.05	0.31
(E, 1000, 0.25)	0.11	0.00	0.00	0.00	0.00	0.02	0.06
(F, 1000, 0.25)	0.00	0.00	0.00	0.00	0.05	0.07	0.24
(G, 1000, 0.25)	0.00	0.00	0.00	0.00	0.00	0.02	0.08
(A, 1000, 0.45)	0.60	0.21	0.22	0.71	0.56	0.97	0.98
(B, 1000, 0.45)	0.01	0.01	0.01	0.06	0.12	0.75	0.65
(C, 1000, 0.45)	0.00	0.01	0.01	0.10	0.12	0.31	0.59
(D, 1000, 0.45)	0.70	0.07	0.31	0.01	0.09	0.05	0.63
(E, 1000, 0.45)	0.21	0.07	0.02	0.00	0.00	0.02	0.07
(F, 1000, 0.45)	0.00	0.00	0.00	0.02	0.00	0.04	0.18
(G, 1000, 0.45)	0.00	0.00	0.00	0.00	0.00	0.02	0.09
(A, 1000, 0.65)	1.00	1.00	1.00	0.97	1.00	1.00	1.00
(B, 1000, 0.65)	0.04	0.02	0.03	0.12	0.16	0.72	0.76
(C, 1000, 0.65)	0.00	0.01	0.01	0.08	0.19	0.52	0.42
(D, 1000, 0.65)	1.00	1.00	1.00	1.00	1.00	1.00	1.00
(E, 1000, 0.65)	1.00	1.00	1.00	1.00	0.99	1.00	1.00
(F, 1000, 0.65)	0.00	0.00	0.00	0.00	0.01	0.08	0.16
(G, 1000, 0.65)	0.00	0.00	0.00	0.00	0.00	0.02	0.10

It is important to note that not all elements of Λ were stored when $n = 50000$ or $n = 100000$ due to memory limitations. During these tests, as many elements of Λ were stored as possible, but if additional values were needed for investigation in CDKPSOLVE, they were recalculated as necessary but not permanently stored. This lead to repeated calculations which worsened algorithmic performance. Hence, the results from these values of n are expected to provide poor results and are likely to not be fully competitive with CPLEX. The only method to address this issue is to increase the availability memory on the computer implementing DKPSOLVE.

4.7 Discussion

The results from Table 9 demonstrate that DKPSOLVE outperforms CPLEX in a majority of the test combinations. Specifically, 84.4% of all test combinations from Table 9 indicate that DKPSOLVE is the preferred solution method as opposed to using CPLEX. (indicated by the bolded test combinations in Table 9). If all of the test combinations with 50000 or 100000 variables are excluded, this performance increases to 94.3%. These results clearly demonstrate DKPSOLVE is preferred for problems where all element of Λ can be stored.

Also shown in Table 9 is that 80.3% of the $U = 100$ test combinations had a majority of their test instances solved faster with DKPSOLVE while 88.4% of the $U = 1000$ test combinations were solved faster with DKPSOLVE. Furthermore, if the test combinations with 50000 or 100000 variables are excluded, 99.1% of the $U = 1000$ test combinations outperformed CPLEX. Hence, DKPSOLVE's performance over CPLEX improved as the coefficient range increased. This result is most likely because larger coefficient ranges imply the elements of Λ are more diverse. Therefore, there are less elements per bin (rounded to the nearest hundredth) for CDKPSOLVE to analyze which improves overall performance. Larger values of U are recommended for future testing to determine if this trend continues.

With respect to coefficient correlations, 84.9% of the test combinations for test Cases B, D, and F had a majority of their test instances solved faster with DKPSOLVE while only 81.0% of the test combinations for test Cases C, E, and G had a majority of their test instances solved faster with DKPSOLVE. Hence, as the coefficient correlation increases, DKPSOLVE performance degrades. The likely cause of this performance is because less coefficient correlation implies less variability in the elements of Λ which requires more values to be tested within each call of CDKPSOLVE. Furthermore, all 100% of the test combinations for test Cases B and C, 82.1% of the test combinations for tests Cases D and E, and 66.7% of the test combinations for test Cases F and G were solved faster with DKPSOLVE than with CPLEX as measured by a pairwise comparison. Hence, DKPSOLVE is recommended for all but the most correlated test combinations. With respect to M, 82.7% of the test combinations where $M = 0.25$, 80.6% of the test combinations where $M = 0.45$, and 89.8% of the test combinations where $M = 0.65$ had a majority of the test instances solved faster with DKPSOLVE than with CPLEX as measured by a pairwise comparison. These results demonstrate that once the knapsack constraint becomes easy to satisfy, the performance of DKPSOLVE improves.

In Table 10 and Table 11, the aforementioned trends observed in Table 9 with respect to n, M, U, and Case are similarly demonstrated which further support the previously discussed findings and conclusions. The additional observation from Table 10 and Table

11 is that DKPSOLVE is more likely to experience outliers with respect to solution time than CPLEX. Specifically, only 72.4% of the test combinations observed instances in which the average solution times from DKPSOLVE outperformed the average solution time from CPLEX. Note this percentage increases to 82.9% if the test combinations with $n = 50000$ and $n = 100000$ are excluded. Comparatively, 84.4% of test combinations observed DKPSOLVE outperforming CPLEX in Table 9 (94.3% of the test combinations excluding $n = 50000$ and $n = 100000$) and 83.3% of the test combinations observed DKPSOLVE outperforming CPLEX with respect to median solution time in Table 11 (90.5% of the test combinations excluding $n = 50000$ and $n = 100000$). Hence, DKPSOLVE is much more likely to experience outliers which negatively affect average solution times in comparison to CPLEX. The most likely cause of these outliers is that some instances are not able to determine a high quality value for P (the lower bound) during the REDUCE procedure. If a loose value for P is observed, than few of the variables will be reduced in that problem and more of the branch-and-bound tree will have to be investigated until this bound can be improved.

The values in Table 12 provide the average *absolute* difference between P at the end of the REDUCE Procedure and the best known solution. The absolute difference is utilized as the performance measure because the optimal objective values, especially for the highly correlated instances, are frequently negative. Test instances whose optimal solutions are close to zero will therefore have a higher *percentage* gap between these two values even though P and the optimal solution are similar. Hence, average absolute difference is used so the results for a given U and n are directly comparable and avoid situations where the optimal objective is near zero. Note that the average *percentage* gap (not shown in any of the provided tables) is 2.01% for $n = 100$, 0.30% for $n = 500$, 0.13% for $n = 1000$, and 0.02% or less for all larger values of n. Hence, stopping the procedure after REDUCE is an extremely effective heuristic according to this measure.

These percentage based results, as well as the absolute difference results shown in Table 12, demonstrate that as n increases, the accuracy of REDUCE improves. With respect to M, the average absolute solution gap of P at the end of the REDUCE phase is 45.5 when $M = 0.25$, 36.4 when $M = 0.45$, and 32.7 when $M = 0.65$. Hence, the performance of the REDUCE phase improves as the feasible region becomes larger. With respect to the different Cases, REDUCE performs worst under high levels of coefficient correlation as the average absolute gap is 96.8 for Case F and G test combinations compared with 20.5 for Case B and C test combinations and 11.6 for Case D and E test combinations. The good performance of the Case D and E test combinations are driven by the $M = 0.65$ test combinations as <u>all</u> of these test instances were optimally solved within the REDUCE phase.

Finally, the results from Table 13 show that 30% of the overall time is spent in the REDUCE phase on average. This further demonstrates the strength of using REDUCE as a heuristic. However, it should be noted that certain test combinations do require higher percentages of the overall time in the REDUCE phase. Specifically, as n grows, the percentage of time spent in REDUCE increases. For instance, when $n = 50000$ or $n = 100000$, the average percentage of the overall time in the REDUCE phase is 40% and 45% respectively while only 23% to 25% of the overall time is spent in the REDUCE phase for $n \leq 1000$. Similarly, for test combinations with Case D and E, an average 44% of the overall time is spent in the REDUCE procedure. This is significantly higher than

Case B and C test combinations where only an average 24% of the overall time is spend in the REDUCE procedure and Case F and G test combinations where only an average 3% of the overall time is spent in the REDUCE procedure. These results, along with those from Table 12, demonstrate that the REDUCE procedure is an extremely effective heuristic for the DKP which I believe to be one of the significant contributions of the work presented in this chapter.

In summary, the DKPSOLVE procedure outperforms CPLEX on all measures for nearly all small to moderate size problem (i.e. $n \leq 10000$) when coefficient correlation is not severe. In all other situations, CPLEX is preferred, but DKPSOLVE is extremely competitive. The only challenge with DKPSOLVE is that it is much more sensitive to outliers which worsens average solution time. However, it is recommended that those seeking to solve any type of DKP implement DKPSOLVE on their systems as different implementation environments or hardware may improve DKPSOLVE's performance.

Furthermore, three avenues of future work are recommended. The first is to implement new test combinations such as using larger values of U to observe if the trend of improving results continues with larger coefficient ranges. Secondly, new implementation and data structure techniques may further improve performance and should be investigated. Finally, it is recommended that new heuristic techniques be investigated to improve IMP, REMREPL, and REMREPL2. If these techniques are improved to find better lower bounds, the time required to solve DKP instances will decrease as more variables can be removed in the REDUCE procedure and the branch-and-bound tree will be smaller in the EXPCORE procedure.

4.8 Conclusion

With respect to mobile retailers, the DKPSOLVE procedure is directly applicable assuming the retailer can model their product mix problem as a DKP. In that case, using DKPSOLVE is highly recommended, especially as it does not require the use of commercial software and can be provided as a ‘black box’ whose inputs can be easily provided by any basic user.

Furthermore, a mobile food retailer may be able to simplify their product mix decision in the case that they cannot initially model it as a DKP. For instance, a retailer may want to require their stocked mix to meet a required profit margin and to exceed a given nutritional threshold. This would require two demand constraints and an MDMKP formulation. However, it may be advisable to instead use a multi-objective approach where one of the demand constraints becomes a second objective function. By testing multiple convex combinations of the two objectives, a solution which is feasible to the initial two demand constraint problem can be identified which is likely to be close to the optimal solution. The advantage of this approach is that it will still provide a high quality solution while utilizing a more accessible solution technique. This type of approach will be shown as one part of the case study detailed in Chapter 7.

In summary, the DKP and the DKPSOLVE procedure can aid a mobile food retailer plan their product mix to make better stocking decision. Furthermore, depending on the goals of the retailer, the DKP can be tailored to increase the profitability of the mobile retailer thereby increasing their economic sustainability as shown by the case study in Chapter 7. The disadvantage of the DKP, in comparison to the MDMKP, is that it places much more stringent requirements on the mobile retailer with respect to what they can include in the DKP model. However, if the mobile retailer is able to utilize the DKP, the DKPSOLVE procedure provides a guaranteed optimal solution faster than using MDMKP solution algorithms and DKPSOLVE does not need commercial software.

CHAPTER 5

AN EXACT SOLUTION ALGORITHM FOR THE MOBILE RETAILER ROUTING PROBLEM

Within this chapter, a solution procedure is provided to determine the optimal routing and scheduling plan for a mobile food retailer. This operational decision is modeled as a CCVRP based on the discussion in Chapter 2. In this problem, it is assumed that the retailer manages any number of vehicles which must start and end their routes at a given depot and the vehicles makes connected cycles through the service network. The possible stop locations and the demand at each location is known when scheduling the vehicles and each vehicle has a homogeneous limit on the demand it can serve. The unique element of this problem is that it is assumed that demand/customers at one location may be willing to travel to a nearby location so long as the distance or time required to travel that distance is below a given threshold. For an urban mobile food retailer, this is a valid assumption as potential service locations for the vehicle may be within close proximity (e.g. within 400 meters) of one another. This is especially relevant for food desert communities as low-income customers are price conscious (Kaufman et al. 1997; Andreyeva et al. 2008) and would be willing to travel small distances assuming the mobile food retailer has grocery items which are priced competitively compared with traditional grocery stores.

Unique solution algorithms are needed for the CCVRP as this type of problem has rarely been addressed. As discussed in Chapter 2, problems similar to the CCVRP have only been solved twice. The first was only able to solve instances up to 35 customers exactly and the second modeled demand as a continuous function. Hence, the latter is not directly applicable to the issues faced by mobile food retailers while the former may be outperformed by an improved algorithm. The purpose of this chapter is to introduce a new exact solution algorithm for the CCVRP. A large set of computational experiments are conducted and the results from these tests are discussed along with recommended uses of the exact solution algorithm for mobile food retailers.

5.1 The set covering CCVRP

The formulation of the CCVRP is as follows. Let the indexed set of n customers and the depot be denoted as $V = \{0,1,2,\dots,n\}$ where 0 denotes the vehicle depot. Let the set of n customers, excluding the depot, be denoted as $V' = V\backslash\{0\}$. For each customer $i \in V'$, let $d_i \geq 0$ indicate the demand which must be serviced and let $b_{ij} \in \{0,1\}$ be the binary indicator if customer i can have their demand serviced from a vehicle visiting customer $j \neq i$. For each pair $(i,j) \in V$, let t_{ij} be the cost of traveling between the two locations. This cost is typically defined by the network or Euclidean distance, but can be altered to factor in costs such as fuel and roadway charges. It is assumed there are K vehicles, indexed as 1 through K, which have to be used and each vehicle is homogenous with a demand capacity of C. Clearly $d_i \leq C$ since any violation makes the problem infeasible.

The CCVRP requires two sets of decision variables. Specifically, let

$$x_{ijk} = \begin{cases} 1 & \text{if vehicle } k \text{ traverses between location } i \text{ and location } j, \\ 0 & \text{otherwise} \end{cases}$$

for each customer pair $(i,j) \in V$ and each vehicle $k \in K$. Additionally, let

$$y_{ik} = \begin{cases} 1 & \text{if customer } i \text{ has demand serviced from vehicle } k, \\ 0 & \text{otherwise} \end{cases}$$

for each customer $i \in V'$ and each vehicle $k \in K$. Given these variables, the standard CCVRP formulation (hereafter referred to as P) seeks to

(P) Minimize: $\sum_{(i,j)\in V}\left(t_{ij}\sum_{k\in K} x_{ijk}\right)$ (5-1)

subject to

$\sum_{j=0}^{i-1} x_{jik} + \sum_{j=i+1}^{n} x_{jik} = \sum_{j=0}^{i-1} x_{ijk} + \sum_{j=i+1}^{n} x_{ijk} \qquad \forall\, i \in V, k \in K,$ (5-2)

$\sum_{j\in V'} x_{0jk} = 1 \qquad \forall\, k \in K,$ (5-3)

$\sum_{(i,j)\in S} x_{ijk} \le |S| - 1 \qquad \forall\, S \subseteq V', k \in K,$ (5-4)

$y_{ik} \le \sum_{j\in V'} b_{ij}\left(\sum_{i'=0}^{j-1} x_{ji'k} + \sum_{i'=j+1}^{n} x_{ji'k}\right) \qquad \forall\, i \in V', k \in K,$ (5-5)

$y_{ik} = \sum_{j=0}^{i-1} x_{ijk} + \sum_{j=i+1}^{n} x_{ijk} \qquad \forall\, i \in V', k \in K,$ (5-6)

$\sum_{i\in V'} d_i y_{ik} \le C \qquad \forall\, k \in K,$ (5-7)

$\sum_{k\in K} y_{ik} \ge 1 \qquad \forall\, i \in V',$ (5-8)

$x_{ijk} \in \{0,1\} \qquad \forall\, (i,j) \in V, k \in K,$ (5-9)

$y_{ik} \in \{0,1\} \qquad \forall\, i \in V', k \in K.$ (5-10)

The goal of P is to minimize the total routing cost of all vehicles as expressed by (5-1). This is constrained by the traditional VRP constraint sets (5-2) through (5-4) which respectively ensures that vehicle flow is conserved, that all vehicles must leave the depot, and that complete routes which omit the depot are not permitted (i.e. subtours are not permitted). Constraint set (5-5) ensures that a vehicle cannot satisfy the demand of a customer unless it stops at another location in the service radius of that customer. Constraint set (5-6) forces any vehicle which stops at a customer to service that customer's demand. This requirement was added to the CCVRP as a customer which has a vehicle visit its location, but not service its demand is unrealistic with respect to typical operating conditions. Applications where this is not true can remove these constraints without heavy modifications to the solution procedure. Constraint sets (5-7) and (5-8) respectively guarantee no vehicle's capacity is exceeded and that all customers are served while constraint sets (5-9) and (5-10) ensure that all variables are binary.

To solve this problem optimally, a column generation technique was implemented. This approach was selected because incorporating the covering mechanic into the route generation procedure would be algorithmically simple and it would be easy to modify the approach to fit the needs of any mobile food retailer such as retailers which cannot satisfy all community demand or retailers who want to penalize sales if they are served at a distance. To use column generation, the CCVRP (or CVRP in the case of the cited literature) is transformed into an equivalent set-covering problem such that each variable represents a feasible vehicle route and covering plan. In the case of the CVRP, each variable only represents a feasible vehicle route. The problem is initially solved over a small set of these routes, but more are generated as needed until an optimal solution is obtained. Within the presented algorithm, routes are generated using a branch-and-bound approach similar to Agarwal, Mathur, and Salkin (1989).

Using this framework to create routes, optimality for the integer problem can be achieved via a branch-and-price methodology. In this approach, some aspect of the problem (e.g. an edge in the network) serves as the branching criteria and a relaxed version of the problem, typically the linear relaxation given the current branching, is solved. During this solution process, more routes are generated as needed. If this relaxation has a higher objective than the current best binary problem, the node is fathomed. The binary problem is solved via branch-and-price once all nodes are fathomed and the best binary solution identified is therefore optimal. This method was made most popular by Desrochers, Desrosiers, and Solomon (1992) who solved a CVRP with time windows via this methodology.

To solve the CCVRP using a column generation approach, P must be transformed into an equivalent set-covering formulation. Let $\mathcal{R} = \{1,2,\ldots,R\}$ represent the *complete* set of feasible routes. These routes must start and end at the depot and included the complete routing and covering plan such that the covering does not exceed C and all customers directly visited on the route are serviced in the covering plan. For all $r \in \mathcal{R}$, let $T_r \subseteq V$ represent the ordered set of customers and the depot along the routing plan for route r and let $S_r \subseteq V'$ represent the customers in the covering plan for route r. Clearly $T_r \backslash \{0\} \subseteq S_r$ due to constraint set (5-6). Let c_r be the cost of route r which is the summation of edge costs t_{ij} for traveling along the cycle indexed by T_r. In addition, define

$$\beta_{ir} = \begin{cases} 1 & \text{if customer } i \text{ is covered by route } r, \\ 0 & \text{otherwise} \end{cases}$$

for each customer $i \in V'$ and each route $r \in \mathcal{R}$ and define

$$z_r = \begin{cases} 1 & \text{if route } r \text{ in the optimal solution,} \\ 0 & \text{otherwise} \end{cases}$$

for each route $r \in \mathcal{R}$.

The goal of the set-covering formulation (hereafter referred to as SC) is to select a set of routes such the total cost of the routes is minimal but all customers have their demand satisfied. The formulation is

(SC) Minimize: $\sum_{r \in \mathcal{R}} c_r z_r$ (5-11)

subject to

$\sum_{r\in\mathcal{R}} \beta_{ir} z_r \geq 1 \qquad \forall\, i \in V',$ (5-12)

$\sum_{r\in\mathcal{R}} z_r = K,$ (5-13)

$z_r \in \{0,1\} \qquad \forall\, r \in \mathcal{R}.$ (5-14)

Constraint set (5-12) of SC ensures that all customers are included at least once among the selected routes, similar to constraint set (5-8) in P, while constraint (5-13) requires all K vehicles to be used for routing. Constraint set (5-14) ensures that all variables are binary. It is clear that SC and P will provide the same optimal solution assuming the triangle inequality holds for all t_{ij}.

The challenge with solving the SC problem is that including all possible routes in $\mathcal{R}$ is impossible. Instead, columns are generated as needed using a branch-and-price approach similar to Agarwal, Mathur, and Salkin (1989) within a branch-and-price tree. This complete process works as follows. Initially, the lower bound of the root node in the branch-and-price tree is calculated as the solution to the linear relaxation of the SC problem. If this solution is binary, the SC problem is optimally solved. Otherwise, a starting binary solution is obtained which serves as the first upper bound in the branch-and-price tree. Furthermore, the cumulative travel along some edge $e \in E$, where E is the set of all edges in the network, must be fractional and the branch-and-price tree commences by branching upon the respective inclusion/exclusion of some fractional edge in the optimal SC solution.

For each node in the branch-and-price tree, the linear relaxation of SC is solved assuming the edge requirements/restrictions are enforced. Hence, denote the linear relaxation problem as $RLSC(E_1, E_0)$ (Relaxed Linear Set Covering) where $E_1 \subseteq E$ and $E_0 \subseteq E$ are the edges in the network which are forced to be included and excluded respectively in the relaxed optimal solution for that node. This notation also applies to the root node but $E_1 = E_0 = \emptyset$. For any node in the branch-and-price tree, $RLSC(E_1, E_0)$ is solved via column generation. This technique is shown in subsection 5.2 for the root node (i.e. no edge restrictions are employed) and the necessary changes to solve this lower bound at other points in the branch-and-price tree are discussed in subsection 5.3. Given the solution to $RLSC(E_1, E_0)$ for any non-root node, the node is fathomed if it is greater than the best SC upper bound solution. Otherwise, branching continues by selecting some non-binary edge assuming the solution is non-binary. If $RLSC(E_1, E_0)$ returns a binary solution for a node, the best binary solution is updated if applicable and the node is then fathomed. The process is complete, and the current best SC solution is the optimal solution when all nodes are fathomed in the branch-and-price tree.

5.2 Linear set covering solution methodology for the CCVRP

Consider the $RLSC(E_1, E_0)$ problem at the root node of the branch-and-price tree which is formulated with objective (5-11) and constraints (5-12) through (5-14) except constraint set (5-14) is relaxed such that each decision variable can take any non-negative real value between zero and one. As previously stated, $E_1 = E_0 = \emptyset$ at the root node which is assumed throughout the remainder of this section. To determine the optimal solution for this problem, a starting set of feasible routes is needed. Let $\mathcal{R}' \subset \mathcal{R}$ be such a subset and let $z(\mathcal{R}', E_1, E_0)$ be the optimal solution of $RLSC(E_1, E_0)$ over subset $\mathcal{R}'$. Additionally, let $\bar{\theta}$ and $\bar{\pi} = \{\bar{\pi}_1, \bar{\pi}_2, \dots, \bar{\pi}_n\}$ be the corresponding optimal dual solutions associated with constraint (5-13) and constraint set (5-12) respectively. Clearly $\bar{\theta}$ is unbounded in sign while all elements of $\bar{\pi}$ must be nonnegative.

Given these values, the goal is to determine if $z(\mathcal{R}', E_1, E_0) = z(\mathcal{R}, E_1, E_0)$. Such a condition holds if all constraints are satisfied in $RLSC(E_1, E_0)$ and its dual equivalent *over all routes* $\mathcal{R}$. For $RLSC(E_1, E_0)$, each constraint is clearly satisfied over all $\mathcal{R}$ if they are satisfied over the subset $\mathcal{R}'$. However, the dual formulation of $RLSC(E_1, E_0)$ defined over all routes $\mathcal{R}$ contains the constraint set

$$\textstyle\sum_{i \in V'} \beta_{ir} \pi_i + \theta \leq c_r \qquad (5\text{-}15)$$

for all $r \in \mathcal{R}$. For all routes $\mathcal{R}'$, (5-15) is clearly satisfied by duality theory. However, there may exist some $r \in R \backslash \mathcal{R}'$ which violates (5-15). If this is the case, then adding the violating route r to $\mathcal{R}'$ may lower the future value of $z(\mathcal{R}', E_1, E_0)$ thereby implying $\mathcal{R}'$ does not necessarily contain all of the routes necessary for the optimal solution of $RLSC(E_1, E_0)$.

Hence, the goal of the column generation procedure is to find a feasible route $r \in R \backslash \mathcal{R}'$ such that the reduced cost of the route $\bar{c}_r = \sum_{i \in V'} \beta_{ir} \bar{\pi}_i + \bar{\theta} - c_r$ is strictly positive given the current solution of $RLSC(E_1, E_0)$ over the route set $\mathcal{R}'$. If such a route is identified, then it is added to $\mathcal{R}'$ and $RLSC(E_1, E_0)$ is resolved over the expanded set of routes. If no such route is identified, than $z(\mathcal{R}', E_1, E_0) = z(\mathcal{R}, E_1, E_0)$ and the optimal solution to the linear SC problem has been identified.

The technique implemented to identify such routes is a branch-and-bound procedure which is guaranteed to find the route with the maximum, strictly-positive reduced cost given the current solution of $RLSC(E_1, E_0)$. The technique employed is similar to that of Agarwal, Mathur, and Salkin (1989), but with modifications which improve the solution procedure. Specifically, the technique branches on the inclusion and exclusion of certain customers being physically visited by the vehicle. Once a leaf node has been reached in the tree (i.e. all customers have been known to be included or excluded), a traveling salesman problem (TSP) and a knapsack problem (KP) can be solved to determine the optimal route and service plan for customers along that route. Since both of these problems are NP-hard, a strong upper-bound calculation procedure is needed for each node such that nodes which will not lead to routes with positive reduced costs can be fathomed. This upper-bound calculation is demonstrated in subsection 5.2.1. The complete column generation procedure is then outlined in subsection 5.2.2.

5.2.1 Column generation upper bound

The objective of the branch-and-bound approach is to develop a route of maximum reduced cost such that it has a feasible with respect to both vehicle capacity and routing. Since branching occurs by including and excluding a particular customer from direct visit by the vehicle, the customers can be partitioned using two methodologies. The first is based on whether or not a customer is in the routing plan of the vehicle. Let $V_1 \subseteq V'$ represent those customers who must be visited based on the branching, let $V_0 \subseteq V'$ represent those customers who cannot be visited based on the branching, and let $V_x = V' \backslash (V_1 \cup V_0)$ represent those customers who have yet to be branched. The second partition is based on whether or not the customer is covered due to the aforementioned sets. Let $D_2 = V_1$ represent customers who must be covered as required in (5-6), let $D_1 \subseteq V' \backslash D_2$ represent customers which are not directly visited by the vehicle but some element of V_1 is within their service radius (i.e. they can be serviced), let $D_0 \subseteq V'$ represent customers whose demand cannot be serviced as all customers in their service radius belong to V_0, and let $D_x = V' \backslash (D_2 \cup D_1 \cup D_0)$ represent customers who are not serviceable from any customer in V_1 but can be serviced by visiting some customer in V_x. Let the two partitions be denoted as $\boldsymbol{V} = \{V_1, V_0, V_x\}$ and $\boldsymbol{D} = \{D_2, D_1, D_0, D_x\}$.

To calculate the upper bound at each node, let $c(S, T)$ represent the shortest tour that at least covers all customers $T \subseteq V'$ and must at least visit all customers $S \subseteq V'$. For any customer $i \notin S$, let $f_i(S) = \min_{j,k \in S}\{t_{ij} + t_{ik} - t_{jk}\}$ and for any customer $i' \notin T$, let $q_{i'}(S) = \min_{j \notin S}\{f_j(S): b_{ij} = 1\}$. It is simple to demonstrate that for any $T \subseteq V'$, any $S \subseteq T$, and any $i \notin T$ that

$$c(S, T \cup \{i\}) \geq c(S, T) + q_i(S) \tag{5-16}$$

as f greedily inserts some location which is not visited into the existing optimal route without regards to the current or future route feasibility and q finds the minimal such insertion such that i is covered.

Given any $N \subseteq V' \backslash T$, a clear extension of (5-16) is

$$c(S, T \cup N) \geq c(S, T) + \textstyle\sum_{i \in N} q_i(S)/|N| \tag{5-17}$$

using the same logic as for (5-16) except all $q_i(S)$ must be divided by $|N|$ to protect against the case where all elements of N are coverable from the same minimal insertion point. Therefore, given any node in the branch-and-bound column generation tree (i.e. any $\boldsymbol{V}$ and $\boldsymbol{D}$) and any $D' \subseteq D_x$, then

$$c(V_1, D_2 \cup D_1 \cup D') \geq c(V_1, D_2 \cup D_1) + \textstyle\sum_{i \in D'} q_i(V_1)/n' \tag{5-18}$$

where n' represents the maximum number of customers which can be added to the covering plan. (5-18) therefore provides a lower bound on the routing cost to service the set of customers D' which have yet to be covered by V_1.

Lastly, define $w_i = \bar{\pi}_i - q_i(V_1)/n'$ for each $i \in D_x$ and let x_j be a binary decision variable for all $i \in D_1 \cup D_x$ such that $x_i = 1$ if that customers demand is serviced and $x_i = 0$ otherwise. An upper bound on the maximum reduced cost at any given node in the column generation branch-and-bound tree is determined by solving the following optimization problem:

(UB) Maximize: $\left[\sum_{i \in D_1} \bar{\pi}_i x_i + \sum_{i \in D_x} w_i x_i\right] + \sum_{i \in D_2} \bar{\pi}_i + \bar{\theta} - c(V_1, D_2 \cup D_1)$ (5-19)

subject to

$\sum_{i \in D_1 \cup D_x} d_i x_i \leq C - \sum_{i \in D_2} d_i,$ (5-20)

$x_i \in \{0,1\} \quad \forall\, i \in D_1 \cup D_x.$ (5-21)

UB is a knapsack problem which seeks to select the customers from D_1 and D_x which maximizes the reduced cost of the route such that the vehicles capacity is not exceeded. Note that UB assumes all D_2 must be included in the route by definition. Clearly any node for which UB is less than or equal to zero can be fathomed as no further branching will lead to a covering route with positive reduced cost. Furthermore, any node for which UB is less than the reduced cost of the best route identified thus far can be fathomed as the principal goal of the column generation procedure is to identify the route with the maximum reduced cost.

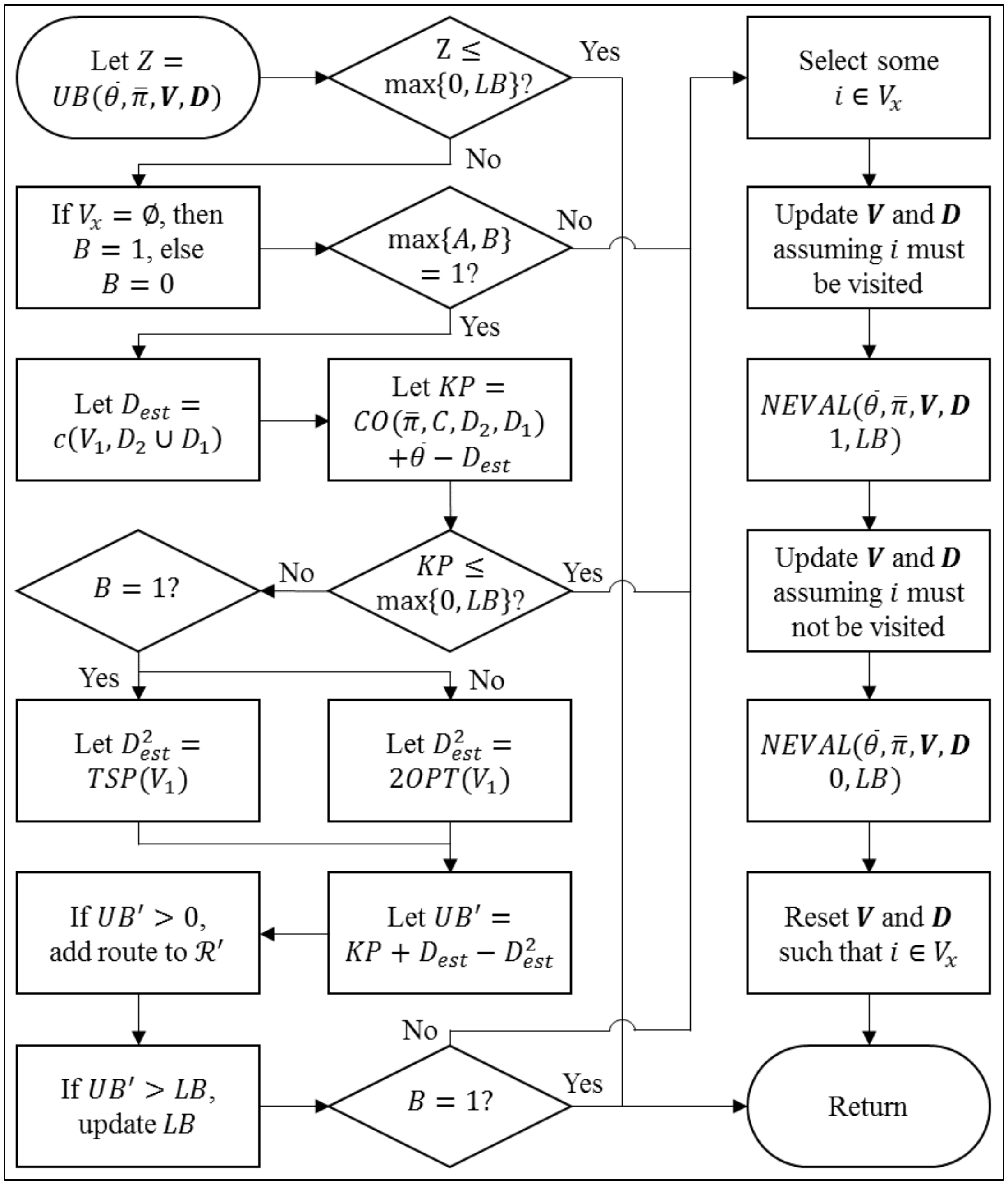
Calculating UB for any node requires the solution of a binary knapsack problem. Even though such a problem is NP-Hard, extremely efficient algorithms exist for solving such problems (Martello, Pisinger, and Toth 1999). Furthermore, since only an upper bound is required, (5-21) can be relaxed to make UB a linear knapsack problem which can be solved extremely efficiently in polynomial time (Balas and Zemel 1980). While such a relaxation results in a looser upper bound, experimentation found this approach is preferred with respect to overall solution time.

Calculating n' at any node is completed by ranking the demands in increasing order for all customers in $D_1 \cup D_x$ and then determining the largest index k such that the sum of the first k values does not exceed $C - \sum_{i \in D_2} d_i x_i$. Calculating $c(V_1, D_2 \cup D_1)$ for any node is significantly harder as it would require solving a TSP problem over $V_1 \cup \{0\}$ as these locations are guaranteed to cover all customers in $D_2 \cup D_1$ by construction. However, it is recommended that this value be estimated from the prior node in the tree. To demonstrate this estimate, let $i \in V'$ be the customer who was branched from the parent node. If $i \in V_1$ in the current node, then the TSP tour can be estimated based on the TSP tour estimate from the parent node plus $f_i(V_1 \backslash \{i\})$. If $i \in V_0$ in the current node, then the TSP tour estimate is the same as that from the parent node. This clearly underestimates $c(V_1, D_2 \cup D_1)$ at a node, but this still allows UB to serve as a valid upper bound.

5.2.2 Branch-and-bound linear procedure

In addition to being able to calculate an upper limit on $\bar{c}_r$ to fathom unpromising nodes, the column generation procedure should also have the ability to add any route with a positive reduced cost, not just the route with maximum reduced cost. Such a capability drastically decreases the solution time of solving $RLSC(E_1, E_0)$ as the computationally intensive column-generation procedure can be called less frequently (Agarwal, Mathur, and Salkin 1989). However, the calculation of UB does not provide a valid covering, since the linear relaxation is solved, nor does it provide a valid routing plan to complete that covering. Since both of these problems are NP-hard (i.e. a binary knapsack problem and a TSP problem respectively), they must solved only when needed, using efficient techniques, and approximated at other times.

To calculate both UB and generates routes as possible, $NEVAL$ (Node Evaluation) is called at every node in the branch-and-bound tree. For input, $\bar{\theta}$ and $\bar{\pi}$ are the current optimal solutions from the dual equivalent of $RLSC(E_1, E_0)$ over the current set of routes $\mathcal{R}'$ and $\boldsymbol{V}$ and $\boldsymbol{D}$ are the two partitions of customers. In addition, let A represent a Boolean input which is true if the current node possibly contains a new route for addition to $\mathcal{R}'$ and false otherwise. Equivalently, if $i \in V'$ is the customer which was branched from the parent node, then $A = 1$ if $i \in V_1$ in the current node and $A = 0$ if $i \in V_0$ in the current node. This logic allows computationally intensive procedures to only be called when a previously unchecked route combination has been encountered. Finally, let LB equal the maximum reduced cost of any feasible route encountered thus far. A graphical summary of $NEVAL$ is given in Figure 6.



Let $Z = UB(\dot{\theta}, \bar{\pi}, \boldsymbol{V}, \boldsymbol{D})$
$Z \leq \max\{0, LB\}$?
Yes
No
Select some $i \in V_x$
If $V_x = \emptyset$, then $B = 1$, else $B = 0$
$\max\{A, B\} = 1$?
No
Yes
Update $\boldsymbol{V}$ and $\boldsymbol{D}$ assuming i must be visited
Let $D_{est} = c(V_1, D_2 \cup D_1)$
Let $KP = CO(\bar{\pi}, C, D_2, D_1) + \dot{\theta} - D_{est}$
$NEVAL(\dot{\theta}, \bar{\pi}, \boldsymbol{V}, \boldsymbol{D}$ $1, LB)$
$B = 1$?
No
$KP \leq \max\{0, LB\}$?
Yes
Update $\boldsymbol{V}$ and $\boldsymbol{D}$ assuming i must not be visited
Yes
No
Let $D_{est}^2 = TSP(V_1)$
Let $D_{est}^2 = 2OPT(V_1)$
$NEVAL(\dot{\theta}, \bar{\pi}, \boldsymbol{V}, \boldsymbol{D}$ $0, LB)$
If $UB' > 0$, add route to $\mathcal{R}'$
Let $UB' = KP + D_{est} - D_{est}^2$
Reset $\boldsymbol{V}$ and $\boldsymbol{D}$ such that $i \in V_x$
No
If $UB' > LB$, update LB
$B = 1$?
Yes
Return

Figure 6. $NEVAL(\bar{\theta}, \bar{\pi}, \boldsymbol{V}, \boldsymbol{D}, A, LB)$ pseudocode flowchart

The first step of $NEVAL$ is to calculate UB and if this value is less than zero or the highest reduced cost observed thus far in the tree, then the node is fathomed as no future branching will ever lead to a route which can be added to $\mathcal{R}'$. Otherwise, a check is performed to see if no future branching will be performed because a leaf node is reached (i.e. $V_x = \emptyset$) and the Boolean value B is set based on this check. If B is true or a new route can be evaluated (i.e. $A = 1$), then a feasible covering and routing plan will be constructed for this node so it can possibly be added to $\mathcal{R}'$.

First, the procedure CO (Combo) is called which solves a binary knapsack problem over D_2 and D_1 where $\bar{\pi}_i$'s are the objective coefficients, d_i's are the knapsack coefficients, and C is the knapsack volume assuming that all customers in D_2 must be selected. The solution of this problem determines the optimal covering if a vehicle were to visit all points in V_1. In this implementation, the binary KP algorithm by Martello, Pisinger, and Toth (1999) was utilized since it is the fastest binary knapsack solver to-date and proved to be extremely efficient in preliminary computational experiments. The solution to CO is combined with $\bar{\theta}$ and $c(V_1, D_2 \cup D_1)$ to calculate KP which provides an upper bound on the reduced cost of a route which only stops at all customers in V_1. Hence, if this upper bound is less than zero or the reduced cost of the current best route, then such a route should not be further evaluated for addition to $\mathcal{R}'$, but the node is not necessarily fathomed as it can still lead to promising nodes in the future.

If the current node is still promising for addition, then all that remains is to determine a feasible route through customers in V_1 such that the route can be added to $\mathcal{R}'$ if its reduced cost is positive. If the current node does not represent a leaf node (i.e. $V_x \neq \emptyset$), then a feasible route is constructed via a two-opt heuristic (Laporte and Semet 2014) as indicated by the function $2OPT(V_1)$. This heuristic is completed since the alternative is to solve a TSP, indicated by the function $TSP(V_1)$, which is optimal, but significantly harder to calculate and is therefore only done when necessary (i.e. when the current node is a leaf node). Once a feasible route distance is found, UB' is calculated which is an updated version of KP except it now has a feasible covering and routing plan. If the reduced cost of the route is strictly positive, it is added to $\mathcal{R}'$. Furthermore, if the reduced cost is greater than LB, then the lower bound is updated.

If the node was not fathomed, the final phase is to continue the branching process by first selecting some $i \in V_x$. For this implementation, i was determined such that it is the most frequently selected, minimal unvisited location (i.e. the argument which provides the minimal value in the function q) for the customers in D_x selected in UB. For instance, if three customers in D_x were non-zero in the solution to UB and location A was the unvisited point with minimum insertion distance which covered two of those customers while location B was the unvisited point with minimum insertion distance covering the third customer, then A would be the next customer selected for branching. This methodology provides a strong branching procedure as it allows the most rewarding elements of D_x to be added to D_1 or D_2 thereby providing a tighter upper bound on the actual reduced cost when calculating UB in future nodes. In the case where no customers from D_x were selected in UB, then $i \in V_x$ with the maximum $\bar{\pi}_i$ is selected for branching.

Given $NEVAL$, the procedure for solving $RLSC(E_1, E_0)$ at the root node of the branch-and-price tree is as follows. First, a set of routes is used to initialize $\mathcal{R}'$. The methodology used to determine these routes is given in subsection 5.4.1. Next, $RLSC(E_1, E_0)$ defined over the current set of routes $\mathcal{R}'$ is solved to obtain the values for $\bar{\pi}$ and $\bar{\theta}$ such that columns (i.e. covering routes with positive reduced cost) are generated. The column generation procedure is started by identifying $i \in V'$ such that $i =$

$\operatorname{argmin}_{i \in V'} \left(\sum_{j \in V', b_{ij}=1} \bar{\pi}_j - 2d_{oi} \right)$ which serves as the first branching option. Once all nodes are fathomed, the current $RLSC(E_1, E_0)$ solution is optimal over all routes $\mathcal{R}$ if no columns were added to $\mathcal{R}'$. Otherwise, $RLSC(E_1, E_0)$ is resolved over the updated set of routes and columns are again generated for inclusion if possible.

5.3 Binary set covering solution methodology for the CCVRP

As stated in subsection 5.1, the binary solution of the SC problem is determined via a branch-and-price tree in which the inclusion/exclusions of edges in E serve as the branching criteria. This technique is similar to the methodology employed by Desrochers, Desrosiers, and Solomon (1992) for the VRP problem with time windows. At each node in the branch-and-price tree, $RLSC(E_1, E_0)$ is solved via column generation where E_1 and E_0 respectively represent the edges which are required and restricted in the optimal routing plan for that node. This technique was demonstrated for the root node of the branch-and-price tree in subsection 5.2 and the changes necessary to solve $RLSC(E_1, E_0)$ at other nodes is as follows.

After $RLSC(E_1, E_0)$ for the root node is solved, a starting binary upper bound solution for the SC problem is determined by solving the SC problem over the final set of routes $\mathcal{R}'$ generated during the root node. To solve this problem, a commercial IP solver is employed and the maximum time allotted is at most five times the time required to solve $RLSC(E_1, E_0)$ at the root node. Computational testing demonstrated that this limit was never reached, but it is recommended to safeguard against instances where lengthy solution times are possible.

Given this binary upper bound, the branching process is started by selecting any edge $e \in E \backslash \{E_1 \cup E_0\}$ in which the cumulative vehicle travel is not integer in the current parent $RLSC(E_1, E_0)$ solution and alternatively adding this edge to E_1 and E_0 and evaluating each child node. The first task in each child node is to prohibit those routes in $\mathcal{R}'$ which violate the new edge requirements. If $e \in E_1$, then any route in $\mathcal{R}'$ which visits either vertex incident to e but does not include e is prohibited from selection as well as any route in $\mathcal{R}'$ which covers any customer vertex incident to e but does not actual visit the vertex. If $e \in E_0$, then any route in $\mathcal{R}'$ which travels along e is prohibited from selection.

$RLSC(E_1, E_0)$ is then solved for any particular non-root node by using the same technique as solving the root node problem in subsection 5.2.2 with only minor modifications. Specifically, the network data is modified such that any customers which are forced to be visited in order by edges in E_1 are treated as one vertex. Hence, some vertices will represent a single customer while other represent a preset path of customers depending on the elements of E_1. Let $V(E_1, E_0)$ represent the set of these paths, including those ‘paths’ which are a single customer. For each pair $(i, j) \in V(E_1, E_0)$, calculate the four distances representing all possible edge distances connecting the start/end of each customer path i and j. Clearly if i and j represent single customers, these four distances are the same. Such increases in data are necessary since some of these edges may be present in E_0 and therefore not permitted in any route. In addition, for any $i \in V(E_1, E_0)$, update the demand of that vertex such that it is the sum of the demand for all customers represented by the customer path i. Also update all four distances between i and any $j \in V(E_1, E_0) \backslash \{i\}$ such that it is the travel distance plus one half of the customer path distance implied by i. For each $e \in E_0$, simply update the distance, if applicable, between the appropriate elements of $V(E_1, E_0)$ containing the vertices incident e to be a large number.

Given these changes, $RLSC(E_1, E_0)$ at any non-root node can be solved by generating columns as needed. Specifically, $RLSC(E_1, E_0)$ is solved over the set of restricted routes in $\mathcal{R}'$ and $\bar{\pi}$ and $\bar{\theta}$ are obtained. Using these prices, the column generation procedure commences by branching on the inclusion/exclusion of customer in $V(E_1, E_0)$ and evaluating each node using a similar procedure to $NEVAL$. Changes to $NEVAL$ include modifying the calculation of $f_i(V_1)$ within UB to determine the minimum insertion distance of each $i \in V_x$ between any two elements of V_1 testing to all possible orientations of the customer path represented by i. Otherwise, calculating UB is the same as shown in subsection 5.2.1. Likewise, CO is unchanged. The sole remaining differences to $NEVAL$ is that both $2OPT$ and TSP must adhere to the limitations imposed by E_1 and E_0. For both of these procedures, all restrictions in E_0 can be enforced by changing the distance for that edge to a large value. For the $2OPT$ procedure, the heuristic can be easily modified to ensure that any edges in E_1 are included by restricting the edges which can be switched. For the TSP procedure, enforcing the requirements of E_1 is more challenging, but the TSP solution methodology in subsection 5.4.3 simplifies this process. If no columns with a positive reduced cost were generated during this modified $NEVAL$ procedure, then the current solution for $RLSC(E_1, E_0)$ is optimal. Otherwise, $RLSC(E_1, E_0)$ is solved over the expanded set of routes, the updated prices are obtained, and new columns are generated for inclusion.

5.4 Algorithmic improvements for solving the CCVRP optimally

While the methodology in subsections 5.2 and 5.3 determine the optimal solution to the CCVRP, numerous algorithmic improvements improve the solution times for solving problem instances. These improvements focus specifically upon generating an initial set of routes, estimating and bounding the pricing values of $\bar{\pi}$ and $\bar{\theta}$, and improving the TSP procedure. For the latter, this includes tracking known subtour constraints as well as known optimal routes.

5.4.1 Initial routes for the CCVRP

To start solving $RLSC(E_1, E_0)$ at the root node of the branch-and-price tree, an initial set of routes is required. To generate these routes, a savings heuristic is employed to group customers together. A discussion on the savings heuristic for TSP is given by Laporte and Semet (2014). For this application, a list of customers pairs $i, j \in V'$ were ordered based on their savings value $s_{ij} = t_{0i} + t_{0j} - t_{ij}$ from largest to smallest. n routes were then created such that each route visits one customer and then returns to the depot. The list of savings values is then analyzed in order and if the two customers are the start/end of current routes, these routes are merged together so long as the route's capacity is not violated. This process is continued until all s_{ij} have been analyzed. If this does not result in at most K routes, the process is restarted but the list of savings values is reordered such that the first value is placed at the end of the list.

Once a suitable grouping of customers has been identified, each subset of customers is passed to the column generation procedure to determine an efficient covering route for the customers in question. Let $\bar{V}$ represent any such grouping. Specifically, the inclusion/exclusion of $i \in \bar{V}$ along a vehicle route is analyzed by a branch-and-bound tree and each node is evaluated by a simplified version of $NEVAL$. These simplifications include updating the calculation of UB such that it no longer includes (5-20) as the customers in $\bar{V}$ will never violate the constraint and changing (5-19) such that it is solely comprised of distance function c as there are no rewards for visiting customers in this procedure. In addition, CO is never called since all customers in $\bar{V}$ can be included by construction. Calls to the TSP procedure are also replaced by simply completing a two-opt heuristic as a near optimal route is sufficient.

5.4.2 Estimating initial prices

As noted by Agarwal, Mathur, and Salkin (1989), the values of $\bar{\pi}$ and $\bar{\theta}$ during the first initial column generation branch-and-bound trees are typically much larger in absolute value than their final values. Therefore, routes generated during these initial column generation procedures are unlikely to be included in the final, optimal solutions. This observation motivates approximating and restricting the values of $\bar{\pi}$ and $\bar{\theta}$ during the start of each column generation phase and slowly relaxing these bounds during each pass of the procedure.

Specifically, let $\pi' = \{\pi'_1, \pi'_2, \ldots, \pi'_n\}$ be the limit on $\bar{\pi}_i$ for each $i \in V'$ and let θ' be the absolute value of the limit on $\bar{\theta}$. The modified $RLSC(E_1, E_0)$ problem, which restricts the value of dual prices is as follows.

Minimize: $\sum_{r\in\mathcal{R}} c_r z_r + \sum_{i=1}^{n} \pi'_i \mu_i + \theta'(\nu_1 + \nu_2)$ (5-22)

subject to

$\sum_{r\in\mathcal{R}} \beta_{ir} z_r + \mu_i \geq 1 \qquad \forall\, i \in V',$ (5-23)

$\sum_{r\in\mathcal{R}} z_r + \nu_1 - \nu_2 = K,$ (5-24)

$$0 \leq z_r \leq 1 \qquad \forall\, r \in \mathcal{R}', \tag{5-25}$$

$$0 \leq \mu_i, \nu_1, \nu_2 \qquad \forall\, i \in V'. \tag{5-26}$$

The modified $RLSC(E_1, E_0)$ defined over a set of routes $\mathcal{R}'$ is solved as a replacement to $RLSC(E_1, E_0)$ in all nodes of the branch-and-price tree from subsections 5.2 and 5.3. Given an initial solution to this modified formulation, the column generation procedure is employed to add any applicable routes. If μ_i for any $i \in V'$, ν_1, or ν_2 are strictly greater than zero in the current solution to the modified $RLSC(E_1, E_0)$, all limits are increased by a multiplier value α and the problem is resolved regardless of the number of columns generated. If columns were generated but all non-route variables were equal to zero, then the problem is resolved with the new routes, but the limits are not increased. If no columns were generated and all non-route variables were equal to zero, then the solution is optimal for $RLSC(E_1, E_0)$. This procedure is identical to the methodology employed by Agarwal, Mathur, and Salkin (1989) to solve the traditional CVRP and similar time savings were observed when solving the CCVRP as was seen for the CVRP.

All that remains is to determine how to set the initial values for the limits π' and θ'. For solving the modified $RLSC(E_1, E_0)$ at the root node, the limits are estimated in a similar manner as recommended by Agarwal, Mathur, and Salkin (1989). In their technique, Agarwal, Mathur, and Salkin estimated these limits by assuming the initial heuristic route solutions (from the savings heuristics in this implementation) will be close to the optimal solution. Hence, the reduced cost of these routes will be zero. Assume that a route $r \in \mathcal{R}'$ from the initial heuristic solution serves customers $\bar{V} \subseteq V'$ and has route cost c_r. Hence, it is assumed that

$$c_r = \textstyle\sum_{i \in \bar{V}} \hat{\pi}_i - \hat{\theta} \tag{5-27}$$

where $\hat{\pi}_i$ is the estimate of the optimal $\bar{\pi}_i$ and $\hat{\theta}$ is the estimate of the optimal $\bar{\theta}$. Each $\hat{\pi}_i$ is therefore estimated as

$$\hat{\pi}_i = \left(1/\left(c_r - \hat{\theta}\right)\right)\left(\beta_1\left(t_{0i}/\sum_{j\in\bar{V}} t_{0j}\right) + \beta_2\left(d_i/\sum_{j\in\bar{V}} d_j\right) + \beta_3\left(q_i/\sum_{j\in\bar{V}} q_j\right)\right) \tag{5-28}$$

where $q_i = \min_{j,k\in V'}\{t_{ji} + t_{ik} - t_{jk}\}$. The initial pricing estimates for a given customer is therefore a linear combination of the depot distance for that customer, the demand for that customer, and the minimum insertion distance from that customer compared with all other customers along that route.

Based on the results observed by Agarwal, Mathur, and Salkin, $\beta_1 = 0.50$, $\beta_2 = 0.40$, and $\beta_3 = 0.10$ are close to the weights which provide the best results for these estimates. In this application, $\hat{\theta}$ was calculated as the average value over all $\hat{\pi}_i$, but future testing could be conducted to determine a better approximation method. To avoid using overly restrictive values, the initial values for π' and θ' were set as 120% of the calculations for $\hat{\pi}_i$ and $\hat{\theta}$. For any $RLSC(E_1, E_0)$ not at the root node of the branch-and-price tree, the initial values for π' and θ' were set as 101% of the final values of $\bar{\pi}$ and $\bar{\theta}$ from the parent node. Since these prices were optimal for a similar problem, it is estimated these limits are an effective starting point for the current problem.

5.4.3 TSP procedure and subtour constraints

The most time-intensive component of evaluating each node in the branch-and-price tree is solving the TSP problem. Hence, an efficient but simple method should be employed to solve the problem. In addition, it should be able to incorporate any edge restrictions/requirements imposed by the branch-and-price tree without significant modification. Furthermore, observe that the TSP procedure will only have to solve shorter route problems as the covering mechanic frequently minimizes the number of stops along a route in comparison to the equivalent VRP.

To demonstrate the TSP procedure employed in this implementation, let $\bar{V} \subseteq V$ represent a set of customers and the depot. Define x'_{ij} for each $i, j \in \bar{V}$ such that $x'_{ij} = 1$ if the vehicle travels from i to j in the solution and $x'_{ij} = 0$ otherwise. The TSP formulation is as follows:

Minimize: $\sum_{(i,j)\in\bar{V}} t_{ij}x'_{ij}$ (5-29)

subject to

$\sum_{j\in\bar{V}\setminus\{i\}} x'_{ij} = 1 \qquad \forall\, i \in \bar{V},$ (5-30)

$\sum_{j\in\bar{V}\setminus\{i\}} x'_{ji} = 1 \qquad \forall\, i \in \bar{V},$ (5-31)

$\sum_{(i,j)\in S} x'_{ij} \le |S| - 1 \qquad \forall\, S \subset \bar{V},$ (5-32)

$x'_{ij} \in \{0,1\} \qquad \forall\, i, j \in \bar{V}.$ (5-33)

As was the case with solving the CCVRP, the challenge with solving the TSP is the subtour elimination constraint set (5-32) which grow exponentially as $|\bar{V}|$ increases. There are multiple techniques to address this problem. On one extreme is to employ complex, but efficient, techniques which utilize combinations of cutting planes and relaxations to solve the binary problem. The most efficient such method is the Concorde solver which has solved the largest TSP problems thus far. An opposing methodology is to directly employ a standard MIP solver and add subtour constraints only as needed. As noted by Pferschy and Staněk (2014), modern MIP solvers have significantly increased in efficiency which makes this approach appealing, especially for smaller problems. Since TSPs in the CCVRP are typically over small sets of customers, this method is preferred due to its simplicity, efficiency, and ability to easily integrate any edge restrictions/requirements.

In this implementation, the latter approach was employed such that (5-29) was solved subject to (5-30), (5-31), and (5-33) over a customer set $\bar{V}$. After the TSP is solved, the solution is scanned for violated subtour constraints. If no subtour is identified, the solution is optimal. Otherwise, the violated subtour constraint(s) is added and the problem is resolved. In addition, a list of violated subtour constraints is saved after each call to the TSP procedure. If future calls of the TSP procedure contain a subset of customers for which a subtour constraint existed, that constraint is added at the start of the TSP procedure. While this requires more initial time to scan the list for applicable subsets of customers, the savings in solving the TSP problem outweighed this search time based on preliminary computational testing.

5.4.4 Optimal TSP routes

The final algorithmic improvement was motivated by the observation that calls to the TSP procedure were frequently over the same subset of customers. Specifically, one instance identified that over 75% of the TSP routes generated were routes which were previously found in prior TSP solutions. Hence, significant computational effort could be saved if known TSP routes were stored and searched prior to calling the TSP procedure.

This storage was implemented by saving both the ordering of customers along that specific route along with any edge restrictions/requirements enforced when constructing the route. This latter storage is necessary during the branch-and-price tree since a customer over the same customer set without those restrictions (i.e. a different place in the tree) may be better served by a shorter, less restricted path. These stored routes are then searched prior to any call to the TSP procedure for both customers and existing restrictions/requirements regarding travel over that subset of customers. If a match is found, the identified route is the optimal route. Otherwise, the TSP procedure is called and the identified route is added to the stored list.

The issue with this approach is that the list of routes can often grow quickly and searching the entire list is time-intensive. Therefore, it is highly recommended that routes which are likely to not be used be removed from the list. Such routes are those with edge restrictions/requirements which are no longer needed as all nodes enforcing such limits have been fathomed. Therefore, if a node is fathomed or all nodes below it are fathomed within the branch-and-price tree from subsection 5.3, the list of stored routes is searched for any route which contains the edge restriction/requirement for that node. If such a route is found, it is removed from the list.

5.5 Computational experiments

Numerous computational experiments were conducted to test the efficiency of the branch-and-price methodology for solving the CCVRP. These tests were completed on CVRP data instances from Set A and Set P by Augerat et al. as well as from Christofides and Eilon for any problem instances which had geographical coordinates for at most 50 customers. Copies of these benchmark cases are maintained by the Networking and Emerging Optimization Research Group (2013). These three data sets were selected as they contained the highest quantity of benchmark cases with up to 50 customers. This customer cutoff was selected as problems with 50 or more customers frequently exceeded memory limitations on the tested hardware. If more customers are to be analyzed, hardware must be increased or techniques to reduce memory usage must be introduced such as not storing subtours or known TSP optimal routes.

In addition to the data provided for each test instance, a coverage policy was established for all customers in each instance. To keep this policy simple, three uniform coverage radii were determined for each problem instance. This was conducted in order to observe the effect of coverage matrix sparsity on solution efficiency. Specifically, three cases were developed (A, B, and C) per problem. Each case is assigned a coverage radius for each test instance based upon the minimum value of the two alternative shown in Table 14. These service radii were applied to each customer such that if customer X is within the service radius of customer Y, then customer Y can be serviced by a vehicle stopping at customer X and vice versa. Clearly test case A has the sparsest coverage matrix while test case C has the densest coverage matrix for each problem instance. These thresholds were developed as they provided a wide range of problems in preliminary testing.

Table 14. Radii criteria for test instances

Case	Single Customer Coverage	Multi-Customer Coverage
A	Max. distance such that at most 1 cust. is covered by 15% of other customers	Min. distance such that at least 75% of customers are covered by at least 1 other cust.
B	Max. distance such that at most 1 cust. is covered by 25% of other customers	Min. distance such that at least 75% of customers are covered by at least 2 other cust.
C	Max. distance such that at most 1 cust. is covered by 35% of other customers	Min. distance such that at least 75% of customers are covered by at least 3 other cust.

All procedures were coded in C++ and all tests were conducted using a single 2.2GHz processor with 8 GB of RAM on a Windows 10 PC. CPLEX v.12 was used as the linear and binary solver. However, UB and CO were solved using a quicksort procedure and the algorithm outlined by Martello, Pisinger, and Toth (1999) respectively. All test instances were terminated after one hour if the branch-and-price procedure was not completed and the best solution identified thus far was recorded as a heuristic solution.

All computational results are shown in Table 15. Specifically, there is a row for each instance and three sets of columns representing the results from test cases A, B, and C. Each test instance is given by a two number combination indicating the number of customers and the number of vehicles respectively. 'LP' indicates how long it took to solve the linear CCVRP in minutes (i.e. the root node solution) while 'BP' indicates how long it took to solve the binary CCVRP in minutes if the problem was solved optimally. The measure 'Igap' represents the percentage difference between the linear solution and the initial integer solution found at the root node. The measure 'Fgap' is the final integrality gap which is the percentage difference between the best integer solution found while evaluating the branch-and-price tree and the objective function value of $RLSC(E_1, E_0)$ for the node closest to the root node which has yet to have both child nodes evaluated. Hence, Fgap is 0 for instances which were solved optimally in 60 minutes. Finally, the measure 'Hgap' is the percentage difference between the heuristic CCVRP from subsection 5.4.1 and the best binary solution when the branch-and-price tree terminates.

Table 15. CCVRP Computational Results (* no linear solution was obtained with one hour)

		A					*B*					*C*				
	Instance	LP	BP	Igap	Fgap	Hgap	LP	BP	Igap	Fgap	Hgap	LP	BP	Igap	Fgap	Hgap
Augerat Set A	32/5	0.3	4.2	1.7	0.0	10.8	0.1	60.0	5.3	4.0	5.0	0.0	60.0	8.4	8.4	3.7
	33/5	0.1	2.4	0.8	0.0	12.1	0.0	0.2	0.2	0.0	16.9	0.0	9.4	4.2	0.0	26.1
	33/6	0.0	17.4	2.4	0.0	4.5	0.0	11.7	1.8	0.0	7.6	0.0	1.9	1.4	0.0	10.7
	34/5	0.1	60.0	5.1	5.1	7.1	0.0	60.0	5.8	5.8	12.1	0.1	19.2	3.1	0.0	21.0
	36/5	1.4	60.0	1.0	1.0	7.1	0.4	60.0	4.2	4.2	3.7	0.4	1.6	1.4	0.0	6.9
	37/5	1.5	60.0	1.0	0.3	7.5	0.6	0.6	0.0	0.0	9.1	0.1	3.0	0.8	0.0	13.3
	37/6	0.2	20.3	0.9	0.0	5.9	0.1	60.0	2.2	2.1	9.1	0.1	0.1	0.0	0.0	16.9
	38/5	0.1	60.0	2.8	2.8	6.7	0.3	7.0	0.9	0.0	9.8	0.1	7.7	3.8	0.0	10.6
	39/5	0.8	40.1	1.0	0.0	15.9	0.9	60.0	1.9	1.9	16.1	0.1	21.2	1.7	0.0	24.9
	39/6	0.4	60.0	3.4	3.4	6.2	0.2	60.0	5.1	3.8	5.6	0.1	0.1	0.0	0.0	15.0
	44/7	0.7	60.0	2.2	2.2	4.4	0.4	0.4	0.0	0.0	3.8	0.1	12.5	2.0	0.0	6.2
	45/6	0.6	60.0	2.9	2.9	11.1	0.4	60.0	4.5	4.5	14.3	0.2	60.0	6.1	6.1	20.1
	45/7	0.2	60.0	2.1	2.1	6.7	0.1	60.0	2.5	2.5	11.6	0.2	45.0	1.5	0.0	14.4
	46/7	0.9	17.8	0.1	0.0	2.7	0.2	39.1	0.8	0.0	8.8	0.1	17.3	1.3	0.0	16.3
	48/7	7.4	60.0	3.1	3.1	4.0	0.9	60.0	3.3	3.3	4.9	0.1	60.0	5.3	5.3	5.3
Augerat Set P	16/8	0.0	0.0	2.3	0.0	12.1	0.0	0.0	0.0	0.0	16.4	0.0	0.0	2.9	0.0	17.0
	19/2	0.1	0.1	0.0	0.0	9.6	0.0	4.3	4.5	0.0	15.7	0.0	0.0	0.0	0.0	16.0
	20/2	0.2	0.2	0.0	0.0	17.7	0.1	3.1	3.3	0.0	22.9	0.0	0.0	0.0	0.0	25.8
	22/8	0.0	0.0	0.5	0.0	50.4	0.0	0.0	0.0	0.0	57.1	0.0	0.1	2.5	0.0	54.9
	40/5	0.6	10.8	0.7	0.0	17.7	0.4	60.0	3.2	3.2	25.1	0.2	60.0	2.2	1.8	15.4
	45/5	1.2	60.0	1.1	1.1	22.0	0.6	60.0	1.8	1.8	22.1	0.7	60.0	1.1	1.1	33.1
	50/7	0.3	60.0	4.1	4.1	4.3	0.2	23.4	1.2	0.0	13.1	0.1	9.0	0.5	0.0	10.4
	50/8	0.3	60.0	3.5	3.5	30.1	0.2	6.4	1.1	0.0	41.0	*	*	*	*	*
Christopfides	22/4	0.0	0.6	5.6	0.0	3.6	0.0	0.0	0.0	0.0	10.8	0.0	0.4	3.7	0.0	6.0
	23/3	19.0	60.0	0.9	0.9	3.6	1.6	1.6	0.0	0.0	4.1	0.1	0.1	0.0	0.0	16.1
	30/3	27.8	60.0	12.1	12.1	30.5	3.2	60.0	12.3	12.3	32.3	0.1	60.0	12.4	12.4	42.4
	30/4	0.2	9.7	1.4	0.0	0.5	0.0	0.5	0.9	0.0	1.3	0.0	2.6	2.3	0.0	2.7
	33/4	*	*	*	*	*	*	*	*	*	*	32.3	60.0	0.8	0.8	1.1

A graphical depiction of the solutions for all three cases of the Christofides and Eilon 30 customer, 4 vehicles test instance is shown in Figure 7. Each panel includes the routing solution indicated by solid lines as well as the covering solution (if the customers is not directly visited) by dashed lines. Note that the customer demands are not shown in the figure. The service radius for each customer is also given in each panel as a reference.

In addition to demonstrating the solution to the CCVRP, Figure 7 also shows that the Christofides and Eilon test instances are not uniformly dispersed over the network. However, the test instances by Augerat are more evenly dispersed. This has a significant effect on the results in Table 15 as the Christofides and Eilon instances tended to be much harder to solve for equal size problems. Therefore, CCVRP problems with uniformly distributed customers appear to be easier to solve than problems whose customers are clustered together.

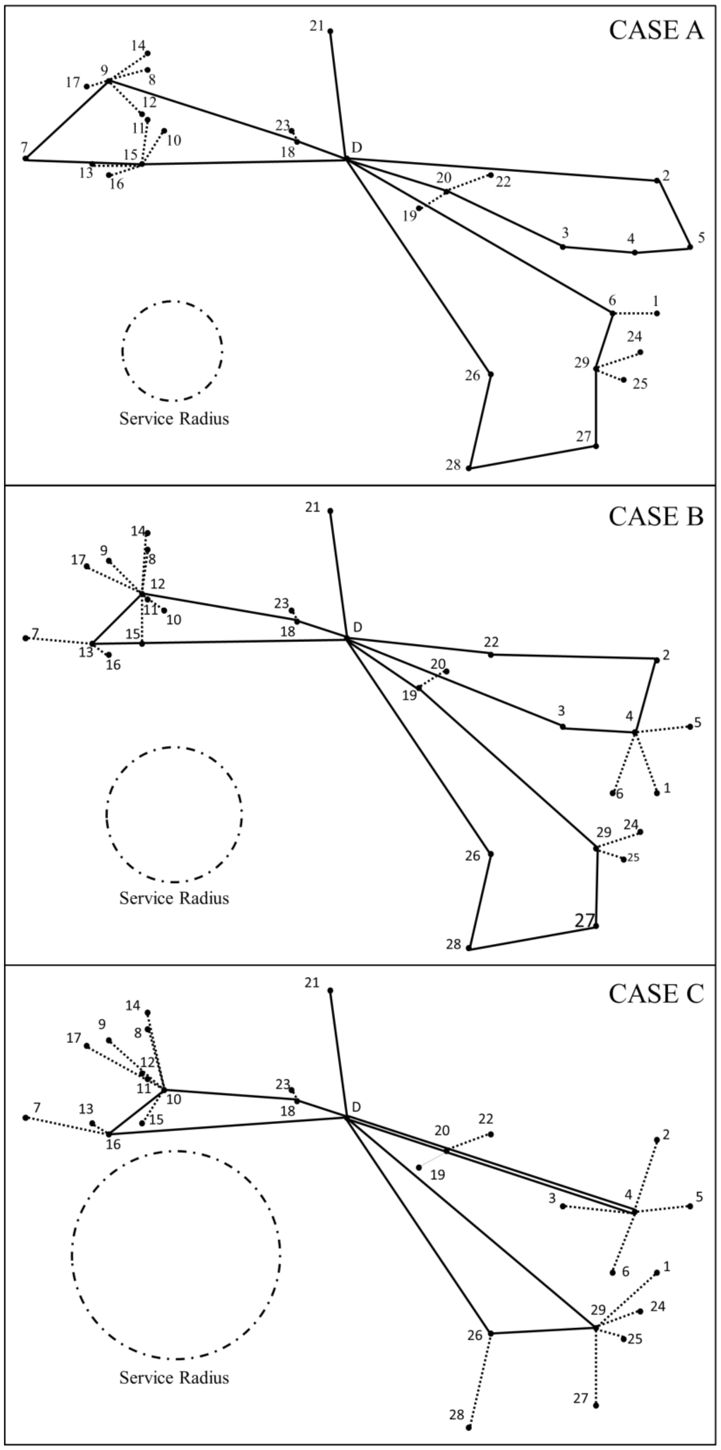


Figure 7. Christofides and Eilon 30 customer, 4 vehicle solutions

The results in Table 15 demonstrate that column generation technique can be used to successively solve many types of problem instances of the CCVRP. Specifically, 48 out of 84 test instances were solved to optimality within the 60 minute time limit. All other problems could be solved given more time to obtain such solutions. Additionally, the column generation method appears more efficient for solving test instances with a denser covering matrix since 13 case A problems were solved optimally, 15 case B problems were solved optimally, and 20 case C problems were solved optimally. The linear relaxation was also solved in all but three of the test instances and many of those solutions, specifically 72 out of 84, were obtained within one minute. Hence, the column generation technique is an efficient solution method whenever the relaxation is needed.

The binary solution obtained after the branch-and-price root node is evaluated, i.e. by solving the integer problem over the columns generated during the solution of the root node problem, has a strong influence on the ability to solve the binary problem. Specifically, the results in Table 16 show the average initial gap for all problems which were optimally solved within one hour compared with the average initial gap for all problems which were not optimally solved with the time limit. Also given in parenthesis are the same averages excluding problems whose linear solution is equal to the binary solution as well as the Christofides and Eilon instance of 30 customers and 3 vehicles as the initial gap in each of these cases is significantly higher than all the others. Regardless of the calculation, problems which were optimally solved within the time limit have a smaller initial gap than those problems which were not solved. While this result is obvious, it indicates that methods which may lower this gap, even at the cost of increased solution time at the start of the branch-and-price process, may be beneficial as it could reduce the overall solution time. This is especially true for problems with a denser coverage matrix as the initial gap for case A test instances is 2.32%, for case B test instances is 2.47%, and for case C test instances is 2.57%.

Table 16. Igap based on case type and one hour solution status

	Case A	Case B	Case C
Test instances solved optimally within 60 min.	1.33% (1.58%)	0.98% (1.63%)	1.66% (2.21%)
Test instances *not* solved optimally within 60 min.	3.24% (2.55%)	4.34% (3.62%)	5.19% (3.98%)

Table 15 also demonstrates that the implemented method is not effective in reducing the gap between the best known binary solution and the linear solution. Specifically, 28 test instances demonstrated no improvement in the gap during the branch-and-price tree. This is principally a result of the branch-and-price process investigating the tree in a depth first manner. This technique was selected since it is easier to implement and can more quickly lead to improved binary solutions. Studies which are interested in lowering this gap should modify this technique such that the branch-and-price tree is investigated in a breadth-first manner.

Finally, the quality of the heuristic savings method utilized to obtain an initial binary solution prior to solving the linear relaxation appears to be a function of the sparsity of the coverage matrix. This is demonstrated by the average gap between the best integer solution and the solution from the savings method adaptation from subsection 5.4.1 which is 11.7% for case A test instances, 14.8% for case B test instances, and 17.4% for case C test instances. Given that the saving method places higher priority on the routing component of the problem, this result is not surprising as a denser coverage matrix implies that the routes will be shorter and less customers will be directly visited. Hence, a different methodology for developing initial routes may be recommended for instances with a dense covering matrix as this will provide a better heuristic solution and may improve the solution times for solving the initial linear relaxation.

5.6 Discussion

The purpose of this chapter was to outline the exact solution procedure for the CCVRP which models the mobile food retailer routing and scheduling problem. The procedure in this chapter utilized a column generation approach which is a new solution method for this CVRP variant. Specifically, the CCVRP was transformed into a set covering problem in which each variable represents a feasible covering route with respect to both vehicle capacity and vehicle routing. This set covering formulation was solved via a branch-and-price methodology where the inclusion/exclusion of edges in the network serve as the branching criteria. At each node in the tree, the linear relaxation of the CCVRP was solved via column generation such that all edge requirements are satisfied. These columns were generated using a branch-and-bound approach. The binary problem was solved once all nodes in the branch-and-price tree were fathomed.

To test the performance of the developed procedure, 84 computational experiments were conducted by employing three difference service radii for 28 test instances. In total, 48 of the experiments were solved to binary optimality within the 60 minute time limit and all but 3 of the 84 experiments were able to have their linear relaxation solved with a majority being solved in less than a minute. The results demonstrate that problems with a denser coverage matrix are easier to solve but the gap between the linear relaxation and the initial binary solution tends to be worse in such problems. Since problems which were not solved had a higher such gap on average, other techniques may be investigated to find an improved, initial binary solution. Furthermore, the heuristic savings solution for case C test instances had a larger gap on average compared with the best binary solution. This implies that other heuristics could be investigated to see if they provide better initial solutions.

Additional research into solving the CCVRP includes investigating alternative solution methods. In this chapter, only a column generation technique was employed. However, other common CVRP solution methods could be investigated including branch-and-cut techniques. The column generation solution method employed in this chapter can also be investigated to see if improvements are possible. Specifically, the branch-and-bound tree from Agarwal, Mathur, and Salkin (1989) was adapted to generate columns, but other techniques are possible including those by Bixby, Coullard, and Simchi-Levi (1997) and Desrochers, Desrosiers, and Solomon (1992) with the latter being extremely successful for most CVRP solutions.

With respect to the employed column generation method, further improvements and tests may be possible. These include testing larger test instances, but the approach must be modified such that hardware limitations are not encountered. Additionally, it may be possible to find a more efficient branch-and-bound procedure by determining a tighter bounding process, specifically with respect to determining the lower bound on the routing cost (i.e. $c(S,T)$) as this value can significantly underestimate the minimum routing distance for certain test instances. Other improvements include utilizing a more efficient

TSP solver, determining a more efficient branching methodology (i.e. which customers should be branched upon when building routes or which edge should be branched upon during the branch-and-price phase), and determining new techniques to identify optimal TSP routes without having to utilize a computationally intensive TSP solver.

5.7 Conclusion

With respect to mobile food retailers. The exact solution algorithm for the CCVRP has limited applicability. First and foremost, the solution requires the use of commercial solvers which limits its wide spread adoption. Second, the exact solution method requires a large time limit to develop optimal solutions, especially for most instances in which there are more than 30 customers. Since most mobile food retailers will have at least this many potential stops, the exact solution method will likely require too much time to identify the true optimal solution. Furthermore, a mobile retailer likely does not require the optimal solution as a routing plan which is near optimal will suffice. Therefore, heuristic solution algorithms are more applicable to the mobile food retailer routing and scheduling problem. While the column generation procedure can be utilized as a heuristic by terminating the procedure early, dedicated CCVRP heuristics are expected to provide more efficient solutions. These heuristic procedures are presented in the next chapter and the results in this chapter serve as a benchmark for solution quality for the developed heuristics.

However, some mobile retailers can utilize an exact CCVRP solution method if they are able to partner with the appropriate practitioners and if their customer network is small enough to be solved within a reasonable time. Under such conditions, the mobile retailer can determine the optimal routes through their customer network such that they minimize their transportation costs thereby increasing their economic sustainability. In addition, the advantage of the column generation procedure is that it is relatively simple to factor in any nuances a mobile retailer may require. For instance, it is relatively easy to modify the problem such that not all customers must be served in the case the retailer does not have enough capacity to satisfy all customer demands. Also, it is easy to factor in penalties for serving customers at a distance to model any losses in profits from customers who may be unwilling to travel if their direct location is not served.

CHAPTER 6

HEURISTIC SOLUTION ALGORITHMS FOR THE MOBILE RETAILER ROUTING PROBLEM

Within this chapter, four heuristic solution procedures are provided for the CCVRP which is equivalent to the mobile food retailer routing and scheduling problem. An exact solution method to address this problem was presented in Chapter 5, but due to the complexity of the problem, it is only practical for smaller test instances. Hence, heuristic solution procedures are necessary to solve problem instances with a realistic number of customers. In particular, four heuristic solution procedures have been developed: a Greedy heuristic, a Sweep heuristic, a Savings heuristic, and an ACS heuristic. The test instances used in Chapter 5 were solved by each of these heuristics in order to observe the differences in solution quality and solution time between the approximate and exact procedures. An additional set of test instances with a larger quantity of customers was also tested to observe the performance of the heuristics for large test instances. The results from these tests and the applicability of the heuristics to the mobile food retailer routing and scheduling problem are discussed.

6.1 Classic routing heuristics

The motivation for adapting classical vehicle routing heuristics such as the Sweep heuristic and the Savings heuristic to the CCVRP is three-fold. First and foremost, numerous commercial applications which requires quick solution algorithms still rely heavily on these techniques as the underlying optimization methodology. Secondly, the CCVRP is much more difficult than the standard VRP as the solution space is much larger due to the covering mechanic. Therefore, the performance of the Greedy, Sweep, and Savings heuristics are unknown and the results from this chapter provide the first documented performance results. Finally, all of these developed heuristics would be easily accessible to mobile food retailers as none require commercial software, in comparison to the methodology developed in Chapter 5, and it is assumed that mobile food retailers would be satisfied with the near optimal solutions generated by these heuristics.

Prior to outlining these heuristics, two key details of the developed procedures must be addressed. First, it is assumed that the formulation for the CCVRP in this chapter is identical to the model defined in (5-1) through (5-10) in Chapter 5. Second, a key element of all the techniques is that they utilize a covering route building procedure which develops a near optimal covering route through a set of locations. A covering route is defined as a cycle through a set of locations such that all locations are visited or the cycle visits a location $j \in V'$ which can service the demand at an unvisited location $i \in V'$ (i.e. $b_{ij} = 1$). This procedure is hereafter called COVROUTE (Covering Route Builder). Note that this is a simplified version of the branch-and-bound procedure $NEVAL$ presented in subsection 5.2.1 and it is identical to the simplification of $NEVAL$ first discussed in subsection 5.4.1. Since COVROUTE is essential to the heuristics in this chapter, it will be explained in detail alongside the pseudocode for the procedure.

Let $S \subseteq V'$ be a subset of customer locations such that $\sum_{i \in S} d_i \leq C$. Hence, including all customers in S on a covering route is guaranteed to not violate the capacity constraint of a vehicle. The goal of COVROUTE is to find a high quality covering route through $S \cup \{0\}$ via a branch-and-bound approach. Specifically, the algorithm branches on the inclusion/exclusion of elements of S along the physical route of the vehicle. To do so, the procedure utilizes sets S_1 and S_0 which are the customers which are included and excluded along the physical route respectively. Additionally, let S^* store the ordered list of physically visited stops in the best route, z^* store the route distance of the best route identified thus far, and z_{LB} track the lower bound on any feasible route through S_1. Hence, initialization starts with S^* and S_0 as empty, $S_1 = \{0\}$ (i.e. must visit the depot), $z^* = \infty$, and $z_{LB} = 0$.

```
**COVROUTE** $(S, S^*, S_1, S_0, z^*, z_{LB})$
    Let $v_i = \min_{j,k \in S_1}\{t_{ji} + t_{ik} - t_{jk}\}$ for all $i \in S \backslash (S_1 \cup S_0)$
    Let $v_i = 0$ for all $i \in S_1$
    Initialize $c_i = \infty$ and let $c_i = \min_{j \in S \backslash S_0}\{v_j \mid b_{ij} = 1\}$ for all $i \in S \backslash S_1$
    *If* $c_i = \infty$ for some $i \in S \backslash S_1$, *then* RETURN
    *If* $\sum_{i \in S \backslash S_1} c_i / |S \backslash S_1| + z_{LB} > z^*$, *then* RETURN
    *If* $\max_{j \in S_1}\{b_{ij}\} = 1$ for all $i \in S \backslash S_1$, *then*
        Let $z$ be a near-optimal tour through $S_1$
        If $z < z^*$, update $z^*$ and $S^*$
    *Else*
        Select some $j^* \in S \backslash (S_1 \cup S_0)$
        COVROUTE $(S, S^*, S_1 \cup \{j^*\}, S_0, z^*, z_{LB} + v_{j^*})$
        COVROUTE $(S, S^*, S_1, S_0 \cup \{j^*\}, z^*, z_{LB})$
    *End if*
```

The COVROUTE procedure starts by calculating the minimum insertion distance v_i for all locations which have yet to be branched. These values assist in providing a lower bound on total tour distance if that location were to be inserted in the tour as demonstrated Chapter 5 and by Agarwal, Mathur, and Salkin (1989). Next, c_i is calculated for any currently uncovered location which represents the minimum insertion distance for some nearby location which can cover location i. Two fathoming tests are then performed. The first ensures that all uncovered stops can still be covered by some unbranched location. The second calculates a lower bound on the current covering route which must at least visit all stops in S_1. This lower bound's validity is demonstrated in Chapter 5. If this lower bound is more than the current best feasible route, the branch can be fathomed.

If the branch cannot be fathomed, two possibilities remain. The first is that all of S is covered by the locations in S_1 in which case no more branching is needed. In such a situation, a cycle is created through S_1 and if this cycle provides a better covering route than the current best route, S^* and z^* are updated. For this implementation, a cycle was created during this phase by employing the 2-opt heuristic as it provided a high quality solution with only a small computational effort. The other possibility is that some location in S has yet to be covered. In this case, some unbranched location j^* must be selected for branching. For this implementation, j^* was selected such that it was the most frequent minimum insertion point that covered the most uncovered locations. In other words, each $j \in S \setminus (S_1 \cup S_0)$ was tallied as the solution for $\operatorname{argmin}_{j \in S \setminus (S_1 \cup S_0)} \{v_i \mid b_{ij} = 1\}$ for all $i \in S'$ where S' are the set of locations which are not currently covered by some location in S_1. This procedure was completed such that solutions were encountered quickly within the branch-and-bound tree in hopes that a high quality upper bound z^* could be identified early in the procedure. Given j^*, the branch-and-bound tree was expanded by including j^* in S_1 and S_0 respectively and the process is continued. The COVROUTE procedure terminates once all paths are fathomed.

6.1.1 Greedy heuristic

The Greedy heuristic is the simplest approach to solving the CCVRP. In summary, the methodology starts with an empty set of covering routes and sequentially builds routes in a greedy manner. A route is started by finding the closest location to the depot which has yet to be serviced and assigning this location as the first stop and assigning it to be covered as required by constraint set (5-6). At this stop, any location which has yet to be served and is coverable by the current stop is assigned to be covered so long as the cumulative capacity served by the route does not exceed C. In this implementation, coverable stops were investigated based on the order of their indices in V. The route is then continued by finding the closest location which has yet to be served such that adding the location to the route does not exceed C. This stop is added to the route and is assumed to be covered. Coverable locations from this new stop are then investigated as previously stated.

This process of building a route is continued until the vehicle seeks to travel to a new location and finds no location exists that can be serviced from the current route without exceeding the vehicle's capacity. The route is therefore considered to be complete and the travel distance from the last stop to the depot is added to the route. The complete list of customers covered by this route is then sent to COVROUTE to determine if an improved route can be identified which services all of these customers. If so, the greedy route is updated. Once this route is completed and assuming all locations have yet to be serviced, a new route is started. This process continues until all locations are serviced and the cumulative sum of all route costs is recorded if K or less routes are used. The process is then repeated by resetting the algorithm and having the first route visit the second closest location first. The routes are then built as previously described. This process of resetting the algorithm is continued after building all routes until the first route visits the farthest point during initialization. The best feasible set of routes from this process is maintained as the heuristic solution.

6.1.2 Sweep heuristic

The Sweep heuristic utilizes a two stage approach. First, V' is partitioned into pairwise independent clusters using the same sweep mechanic as employed in traditional VRPs. Specifically, a location in $i \in V'$ is selected as the start of the sweep and is added to the first cluster. A graphical ray which is fixed at the depot is then swept clockwise starting at i. As the ray encounters a new location, it is added to the current cluster if the cumulative demand of all locations in that cluster does not exceed the current capacity of the vehicle. If demand is exceeded, the current cluster is complete and the current location starts the next cluster. The ray is swept clockwise until all locations are added to clusters. If this creates K or less clusters, than the set of stops in each cluster is given to COVROUTE and a set of feasible covering routes are returned as the heuristic solution.

In this implementation, the Sweep heuristic is conducted n times such that each location in V serves as the initialization point of the heuristic. The best set of routes out of these n options is returned as the solution. Note that it may be possible the Sweep heuristic identifies no feasible set of routes if none of the n initialization criteria are able to partition V into K or less clusters.

6.1.3 Savings heuristic

The Savings heuristic is similar to the Sweep heuristic for the CCVRP in that a two stage approach is used where the first stage partitions V' into pairwise independent clusters. However, this stage utilizes the Savings heuristic which is applied to traditional

VRPs to generate service routes. Specifically, for each pair $(i, j) \in V'$ where $i \neq j$, calculate $s_{ij} = t_{oi} + t_{oj} - t_{ij}$ which is the savings value if two separate routes, assuming i and j were either the first or last customer visited, were joined into one route. Next, sort all s_{ij} in a decreasing manner and create an initial set of n routes such that each route only visits one unique customer in V' and returns to the depot. The list of sorted s_{ij} are then analyzed in order as follows. For any s_{ij}, if i is either the first or last customer visited in one of the current routes and if j is either the first or last customer visited in a different route, then those two routes are merged together maintaining the proper ordering of stops (assuming the newly created route will not exceed the capacity threshold of the vehicle). This process is complete once all s_{ij} have been investigated and the final set of routes is returned. If this process terminates with K or fewer routes, the set of customers in each route are passed to COVROUTE and a set of feasible covering routes are returned as the heuristic solution.

To keep the Savings heuristic approach equivalent to the prior heuristics, the Savings algorithm is conducted n times. The first pass is conducted as described previously. Each subsequent pass resorts the list of s_{ij} such that the first element in the prior call to the Savings heuristic becomes the last element in the current call and all others elements are moved one entry forward in the list. Hence, each call of the heuristic can return a unique set of covering routes and the set with the shortest distance is retained as the solution.

6.2 Ant colony heuristic

The final heuristic is an adaptation of the ACS metaheuristic applied to the CCVRP. The motivation to adapt the ACS procedure for the CCVRP, in comparison to the other metaheuristic approaches, is based on the use of the route building procedure COVROUTE which was developed to be used in all of the presented CCVRP heuristics. The advantage of COVROUTE is that it is able to efficiently consider all possible combinations of visiting/covering locations for building a route through a set of demand points. Its disadvantage is that the algorithm, when performed repeatedly as will be the case in many metaheuristics, is computationally complex when the ratio of customers per truck increases as demonstrated by the computational results in subsection 6.3.3. The ACS procedure was therefore selected as it will require less calls to COVROUTE in comparison to other metaheuristics such as a Tabu Search or Simulated Annealing procedure which may call COVROUTE to measure every insertion/deletion operation. Hence, the discussion that follows solely focuses on the use of the ACS for the CCVRP. The feasibility and use of COVROUTE, as well as the adaptation of other metaheuristics to solve CCVRP instances, is revisited in subsection 6.4.

To simulate the foraging behavior of ants in the ACS, let each ant represent a set of vehicles. Each ant then starts at the depot (i.e. the colony) and moves through the network until all demand is satisfied. This path is then transformed to represent feasible vehicle routes by forcing the ant to visit the depot if the cumulative demand of serviced locations since the ant's last visit to the depot would exceed a vehicle's capacity. To demonstrate how these paths are built, let τ_{ij} for each pair $(i, j) \in V$ represent the amount of current pheromone on the edge connecting those two locations. Furthermore, let $\dot{V} \subseteq V'$ represent all of the current customers who have yet to have their demand serviced and let $\ddot{V} \subseteq \dot{V}$ represent the final subset of visitable customers. The method used to construct $\ddot{V}$ is discussed at the end of this section.

Assuming the ant is currently at location i, the ant can select the next location as

$$j = \arg\max_{j \in \ddot{V}} \left\{ (\tau_{ij})(\eta_{ij})^{\beta} \right\} \quad if\ q \leq q_0 \tag{6-1}$$

where η_{ij} is the inverse of the distance between i and j, $\beta > 0$ is a user defined parameter defining the importance of distance over pheromone strength, $q_0 \in [0,1]$ is a user defined parameter, and $q \in [0,1]$ is a random variable which is defined each time the ant seeks to travel to a new location. If $q > q_0$, then the ant travels from i to a random location in $\ddot{V}$ where the probability that $j \in \ddot{V}$ is selected is

$$(\tau_{ij})(\eta_{ij})^{\beta} / \left(\sum_{k \in \ddot{V}} (\tau_{ik})(\eta_{ik})^{\beta}\right). \tag{6-2}$$

Hence, by adjusting q_0, the ant can either have more emphasis on selecting a random location as expressed by probability (6-2) or more emphasis on selecting the 'closest' location as measured by (6-1). If traveling to the next location j would exceed the current capacity of a vehicle, the current amount of demand serviced by the ant is reset to zero to represent the vehicle returning to the depot before traveling to j.

The novel aspect of the ACS procedure applied to the CCVRP is how the covering mechanism is modeled within the ACS framework. Specifically, a second pheromone mechanic is employed which is separate from the pheromones used to model the physical travel of the ants. To demonstrate this procedure, define the covering pheromone $\tau_{ij}^{c} \in [0,1]$ for each pair $(i.j) \in V'$ which represents the probability that an ant visiting i will service the demand at location j. To ensure only feasible coverings occur, define $\tau_{ij}^{c} = 0$ for any (i,j) pair where $b_{ij} = 0$. Hence, each time an ant visits a new location, all unserved coverable locations are sequentially tested for coverage based on τ_{ij}^{c} so long as servicing that location will not exceed the capacity of that vehicle route. If servicing such a location would exceed the capacity of the current vehicle route, then that location is not investigated for coverage from the current stop.

Given these mechanisms, define an ACS phase as the complete processing of N ants. Hence, a phase starts by spawning N ants at the depot. One at a time, each ant is allowed to create a covering path which is then split into a set of vehicle covering routes such that the vehicle returned to the depot whenever the cumulative demand served along the ant's path exceeded the vehicle's capacity. At the end of each phase, the set of N' ants whose vehicle routes have the shortest total distance are retained as the best and each of their routes are tested for improvement using COVROUTE. If one of these improved ant paths results in K or less vehicle routes whose total distance is less than the best solution found thus far, then the global best is updated. All ants are then reset. In total, the ACS algorithm continues until M total phases have been completed. The routes defining the global best solution at this point are then returned as the final solution.

The remaining key detail of the ACS approach is the updating of pheromones between the phases. Once a phase is completed, pheromones are first evaporated to represent the natural dissipation of scent. For the travel pheromones, evaporate the pheromones as

$$\tau_{ij} = (1-\alpha)\tau_{ij} + (\alpha)\bar{\tau} \tag{6-3}$$

where $\alpha \in [0,1]$ is a user defined value determining the rate of pheromone dissipation and $\bar{\tau}$ is the initial pheromone for each edge in the network. Evaporate the covering pheromones as

$$\tau_{ij}^{c} = (1-\alpha^{c})\tau_{ij}^{c} + (\alpha^{c})\bar{\tau}_{ij}^{c} \tag{6-4}$$

where $\alpha^c \in [0,1]$ is a user defined value determining the rate of covering pheromone dissipation and $\bar{\tau}_{ij}^c$ is the initial covering pheromone which is unique to each edge in the network. After all pheromones are evaporated, those edges which define the best ant path and covering plan found during that ACS phase are updated. Specifically, if edge $(i,j) \in V$ are traveled by the best ant from the phase, then let

$$\tau_{ij} = (1-\alpha)\tau_{ij} + (\alpha)L^{-1} \tag{6-5}$$

where L is the total distance covered by the vehicles which are represented by the ant's path. With respect to the covering pheromones, for any $j \in V'$ which is covered by the best ant stopping at any $i \in V'$, let

$$\tau_{ij}^c = \begin{cases} (1-\alpha^c)\tau_{ij}^c + (\alpha)2\bar{\tau}_{ij}^c & \text{if } L \leq \bar{L} \\ (1-\alpha^c)\tau_{ij}^c & \text{otherwise} \end{cases} \tag{6-6}$$

where $\bar{L}$ is the distance of the best known set of covering routes at the start of the ACS algorithm. By defining $\bar{\tau} = \bar{L}^{-1}$, observe that both (6-5) and (6-6) increase pheromones associated with the best ant path and covering plan only if that plan is better than the best non-ACS route while the pheromones are decreased otherwise. This is advantageous as it protects against rewarding ants that find vehicle routes which are worse than the best known solution at the start of the ACS procedure.

Furthermore, (6-6) is partially motivated by the initialization of $\bar{\tau}_{ij}^c$. For each $j \in V'$, let $f_j = \sum_{i \in V'} b_{ij}$ which indicates the number of locations for which it is possible to satisfy the demand of j. This implementation of the ACS then initializes all $\bar{\tau}_{ij}^c = 1/f_j$ for any $(i,j) \in V'$ pair where $b_{ij} = 1$. Hence, a location has an equal probability of being serviced from any location at the start of the ACS procedure. Additionally, if a location can have its demand serviced from a nearby location (i.e. it can be covered and does not have to be directly visited), then the pheromones in (6-6) will never exceed 1.0 as the highest value for $\bar{\tau}_{ij}^{c}$ is $1/2$.

Even though the prior description suffices to completely define the ACS procedure, one algorithmic enhancement was included to improve the results. Specifically, the subset $\ddot{V}$ used for (6-1) and (6-2) was restricted to only be the closest $(n+1)/7$ unserved locations to the current position of the ant. This improvement is motivated by Bell and McMullen (2004) who implemented the same technique for the ACS applied to the CVRP. Bell and McMullen tested various denominators for determining the size of $\ddot{V}$ with the general results finding larger denominator values (up to 9.0 was tested) improved the results for two of the cases while a denominator of 5 was the best choice in the other case. Hence, a value of 7 was selected as it was only slightly worse than the best in all of the test cases and is therefore a good compromise. Further research could investigate if a different value is preferred, but preliminary testing on some of the test cases in subsection 6.3 identified restricting the size of $\ddot{V}$ improved the quality of the results.

6.3 Computational tests

To test the developed heuristics as well as to perform tuning experiments on the developed ACS procedure, a set of existing CVRP benchmark test instances were expanded to incorporate the covering concept. Specifically, all of set A and set P test instances from Augerat and all test instances from Christofides and Eilon with coordinate data were selected for testing and all customer locations, demands, and vehicle quantities were unchanged. Copies of these benchmark cases are maintained by the Networking and

Emerging Optimization Research Group (2013). These not only include the test instances used in subsection 5.5, but also the test instances which were excluded in Chapter 5 due to the number of customers. For each of these instances, the definitions used to generate the radius cases described in Table 14 were employed.

For these test instances, the computational results are split into two sections based on problem size. The results from problems with strictly less than 50 customers will be presented first followed by the remaining results from larger test instances. This split is necessary since optimal solutions exist only for the smaller test instances based on the results from Chapter 5. Hence, the first set of results can be compared with the known best results while the second set of results can only be compared against the other heuristics in this chapter. Prior to showing these results, parameter setting for the ACS heuristic will be discussed.

6.3.1 ACS heuristic tuning

For this implementation, some of the ACS parameters were assumed to be fixed based on the results identified in the existing literature. Specifically, $N = 25$ ants were simulated at each phase of the ACS procedure and N' was set to 5 as similarly implemented by Bell and McMullen (2004). Increasing N or N' will only improve the results, but it will also slow the procedure. Future research is recommended to study the effect of changing N or N' to find an ideal tradeoff. Additionally, the size of $\ddot{V}$ was fixed based on the discussion at the end of subsection 6.2.

To determine the remaining parameters for the ACS heuristic, two experiments were conducted. In those experiments, three problem instances were selected for testing. Those instances are the 32 location, 5 vehicle case from Augerat's set A, the 51 location, 5 vehicle case from Christofides and Eilon, and the 76 location, 4 vehicle case from Augerat's set P. These three instances were selected as they represent one case from each of the three sets of data and three levels of locations which are reasonable samples from the set of full test instances (which range from 16 to 101 locations). For each of these instances, all three radius cases were tested from Table 14.

A response surface experimental design was conducted to establish the values of α, α^c, β, and q_0. For these experiments it is assumed $M = 5000$ which is revisited later. Initially a four factor (one for each tested parameter) full factorial central composite design was created with $0 \leq \alpha, \alpha^c, q_0 \leq 1$, $1 \leq \beta \leq 5$, axial runs on the edges of the cube, and a center run with one duplication (Montgomery 2012). The initial settings for the four parameters are based on the findings from Bell and McMullen (2004). This experiment has 26 parameter combination settings and each of the 9 test instance and radius combinations were solved 10 times at each setting combination to minimize the effect of outliers. For each of the 9 test instance and radius combinations, the average solution value over the 10 obtained solutions was calculated for each parameter combination setting and the lowest observed solution value was recorded regardless of the parameter settings. The average percentage above the minimum observed value was then calculated and averaged across all 9 of the test instance and radius combinations to determine the singular output from these experiments for each of the parameter combination settings (see Table 17 for an example).

The results from this initial experiment identified that the minimum value was well outside of this experimental region. A steepest descent methodology was employed to move the experimental region until decreasing values were no longer observed. An identical experiment was then conducted over this new region with $0 \leq \alpha, \alpha^c, q_0 \leq 1$ and $8 \leq \beta \leq 12$. The results from these tests are shown in Table 17 where the average percentage above the minimum observed solution value is calculated for each test instance, radius, and parameter setting combination. The average of these values for each parameter setting combination (shown in the right-most column) were the final values used to determine the ideal settings for the four parameters by fitting a full second order response surface model. The minimum value from this model identified the ideal settings to be approximately (rounded for convenience) $\alpha^c = 0.6$, $q_0 = 0.2$, and $\beta = 11$. Within the model, no terms containing α were identified as significant so α was set the same as α^c for convenience. Confirmation experiments were conducted at these settings which identified the average percentage above the minimum observed value as 1.20% which is within the predicted tolerances. If such thorough tuning experiments are not feasible for future practitioners, general tuning procedures can be performed such as Irace or SMAC.

Table 17. ACS tuning parameter designed experiments results for α, α^c, q_0, and β. All results reported as the average percentage deviation from the best identified solution for each instance and radius combination.

Run	$(\alpha, \alpha^c, q_0, \beta)$	Augerat set A			Christofides and Eilon			Augerat set P			**Ave.**
		A	B	C	A	B	C	A	B	C	
1	(0,0,0,8)	0.01	0.02	1.35	2.25	1.73	0.3	2.2	1.09	2.72	**1.30**
2	(0,0,0,12)	0.01	0.00	1.35	2.03	1.83	0.19	2.29	1.29	2.93	**1.33**
3	(0,0,1,8)	8.59	3.93	1.59	10.28	2.78	0.81	4.33	1.59	2.84	**4.08**
4	(0,0,1,12)	8.59	3.93	1.59	10.28	3.09	0.81	4.03	1.11	2.43	**3.98**
5	(0,0.5,0.5,10)	0.02	0.04	1.37	3.18	1.26	0.50	2.74	1.00	2.53	**1.40**
6	(0,1,0,8)	0.01	0.00	1.52	2.08	1.08	0.24	2.38	1.68	2.92	**1.32**
7	(0,1,0,12)	0.01	0.04	1.59	2.06	0.87	0.30	2.10	1.05	1.95	**1.11**
8	(0,1,1,8)	8.59	3.93	1.8	10.28	2.07	0.52	3.96	2.11	2.17	**3.94**
9	(0,1,1,12)	8.59	3.93	2.72	10.28	2.03	0.58	4.18	1.40	2.31	**4.00**
10	(0.5,0,0.5,10)	0.04	0.02	1.35	3.18	2.42	0.59	2.78	1.53	2.80	**1.64**
11	(0.5,0.5,0,10)	0.01	0.02	1.35	1.84	1.24	0.29	2.09	1.16	2.56	**1.17**
12	(0.5,0.5,0.5,8)	0.02	0.06	1.35	2.69	1.35	0.33	3.04	1.32	2.43	**1.40**
13	(0.5,0.5,0.5,10)	0.02	0.04	1.35	3.32	1.34	0.38	2.38	1.36	2.02	**1.36**
14	(0.5,0.5,0.5,10)	0.01	0.06	1.35	2.82	1.39	0.46	2.56	1.01	2.87	**1.39**
15	(0.5,0.5,0.5,12)	0.02	0.02	1.35	2.58	0.89	0.17	2.94	1.45	2.55	**1.33**
16	(0.5,0.5,1,10)	8.59	3.93	1.62	10.28	2.39	0.64	3.86	1.15	2.35	**3.87**
17	(0.5,1,0.5,10)	0.05	0.11	1.59	3.11	1.53	0.48	2.73	1.32	2.60	**1.50**
18	(1,0,0,8)	0.02	0.00	1.35	1.66	1.71	0.31	1.85	1.43	2.85	**1.24**
19	(1,0,0,12)	0.01	0.00	1.35	1.94	1.57	0.51	2.40	0.93	2.42	**1.24**
20	(1,0,1,8)	8.59	3.93	1.59	10.28	3.24	0.67	3.93	1.20	2.20	**3.96**
21	(1,0,1,12)	8.59	3.93	1.59	10.28	2.92	0.79	3.88	1.19	3.20	**4.04**
22	(1,0.5,0.5,10)	0.02	0.06	1.35	3.17	1.25	0.31	2.49	1.10	2.93	**1.41**
23	(1,1,0,8)	0.02	0.00	1.47	1.71	1.00	0.72	3.22	1.24	3.18	**1.39**
24	(1,1,0,12)	0.02	0.02	1.49	1.99	1.38	0.25	1.85	0.83	2.34	**1.13**
25	(1,1,1,8)	8.59	3.93	3.12	10.28	1.88	0.73	4.11	1.29	2.32	**4.03**
26	(1,1,1,12)	8.59	3.93	2.15	10.28	1.85	0.67	4.25	1.47	2.14	**3.92**
Ave.		**2.98**	**1.38**	**1.60**	**5.16**	**1.77**	**0.48**	**3.02**	**1.28**	**2.56**	

Next the value for M was determined given these settings. This was excluded from the prior experiment as the ideal setting for M which provides the lowest solution value is impossible to determine as larger values of M are always expected to improve solution quality. Hence, the goal of determining the ideal setting for M is to identify a balance between solution quality and solution time which cannot be established from a designed experiment. For each of the 9 test instance and radius combinations, values for M from 100 to 10,000 were tested and each was solved 10 times to minimize the effect of outliers. The minimum observed solutions for each of the 9 test instance and radius combinations were recorded from these experiments and the average percentage above this minimum was calculated for each combination of M, test instance, and radius combination.

These values are graphed in Figure 8 and various averages of these values are shown in Figure 9 which were used to identify two key settings for M. The first setting is $M = 4000$ which is the setting prior to the first increase in the overall average percentage over the observed minimum (from 1.07% to 1.19% when $M = 5000$). Hence, $M = 4000$ will be used as it represents a reasonable balance between solution quality and time. The second setting is $M = 7000$ which is the first instance where the lowest overall average percentage over the minimum (0.86%) is observed. Hence $M = 7000$ will also be tested as this represents the scenario where algorithmic time is not an issue.

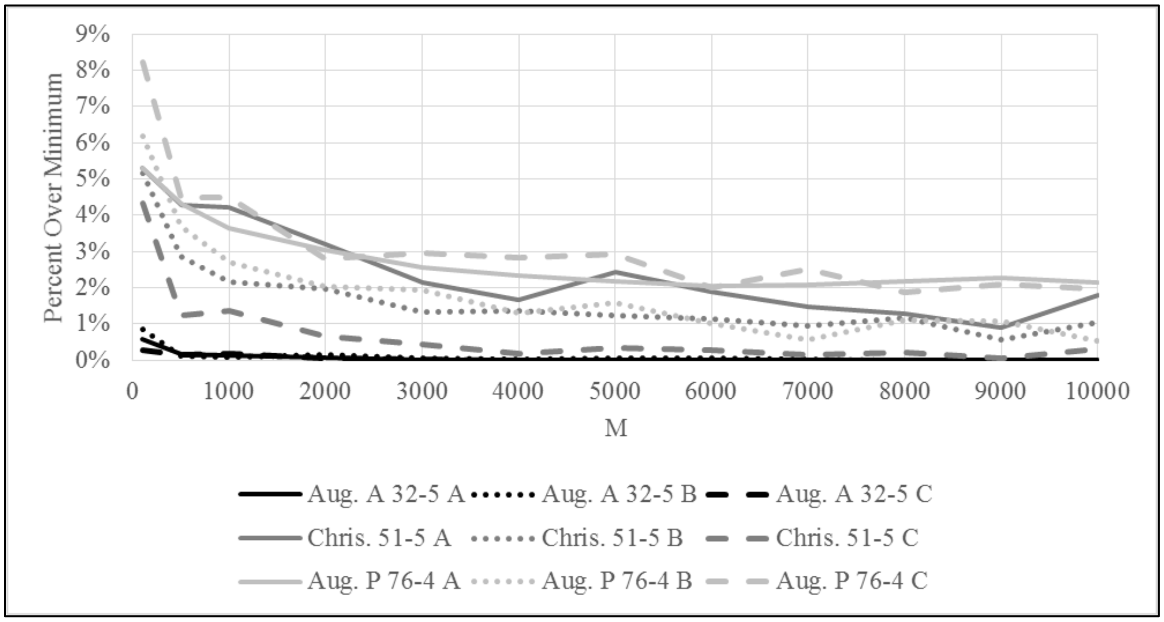


Figure 8. Average deviation from the minimal observed solution for each M, instance, and radius combination.

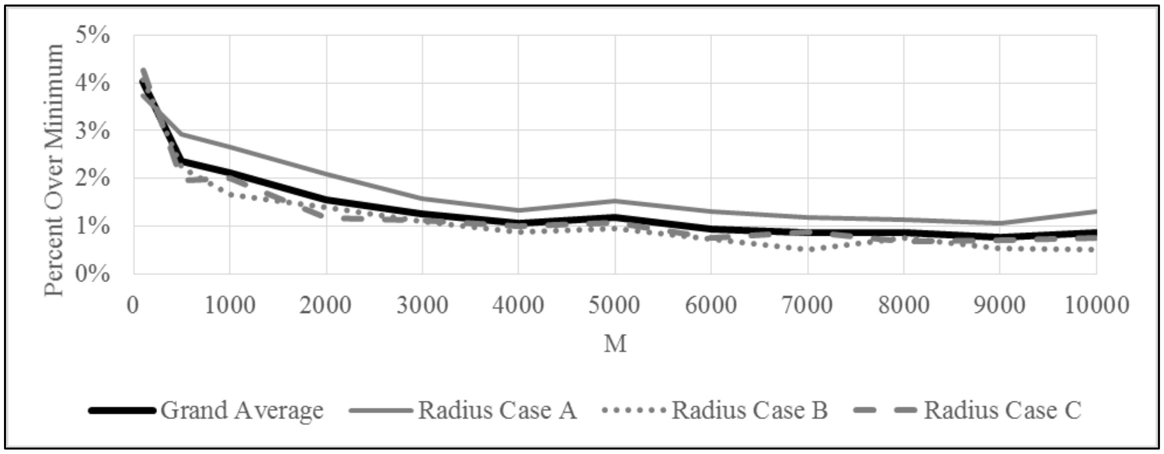


Figure 9. Average deviation from the minimal observed solution across all instances and radius combinations for each M and average deviation from the minimal observed solution across all instances.

6.3.2 Small test instances

Given these settings, computational experiments were conducted on all test instances from Set A and Set P by Augerat as well as all test instances from Christofides and Eilon with strictly less than 50 customers. The results from the 32 location, 5 vehicle Augerat's set A test instance are excluded to avoid overtuning. For each of these test instances, three radius cases were created based on Table 14.

The solution quality and solution time results are reported in Table 18. Specifically, each column in Table 18 reports the indicated statistic for a test case and radius combination. The first row indicates the number of test instances for each set of test combinations. The next set of rows shows the count of test instances in which the heuristic was able to identify a feasible solution. The results from the ACS approach when $M = 4000$ and $M = 7000$ are denoted in the table as ACS4000 and ACS7000 respectively. All reference to the ACS results in the discussion to follow will refer to the ACS7000 results unless otherwise specified. Next are the average ratios of the heuristic solution values over the solution values from the exact solution method from Chapter 5. A subset of this average is shown in parenthesis for only those instances in which the optimal solution value is guaranteed to be identified by the algorithm in Chapter 5. The next set of rows shows the count of test instances in which the heuristic found the best solution value compared with all other heuristics while the final set of rows provides the average and maximum solution times in seconds.

Table 18. Solution quality and solution time results from test instances with strictly less than 50 customers

Measure	Method	Augerat Set A			Augerat Set P			Christofides and Eilon		
		A	B	C	A	B	C	A	B	C
Test Instances	N/A	14	14	14	8	8	8	5	5	5
Count of Feasible Solution Identified	Greedy	14	14	14	8	8	8	5	5	5
	Sweep	12	12	12	6	6	6	5	5	5
	Savings	14	14	14	7	7	7	4	4	4
	ACS4000	14	14	14	7	8	8	5	5	5
	ACS7000	14	14	14	7	8	8	5	5	5
Ave. Heuristic Sol. over Exact Sol. (Only Opt. Sols)	Greedy	1.23 (1.22)	1.26 (1.30)	1.31 (1.32)	1.15 (1.12)	1.18 (1.16)	1.16 (1.18)	1.11 (1.14)	1.13 (1.17)	1.15 (1.20)
	Sweep	1.14 (1.12)	1.18 (1.20)	1.24 (1.24)	1.10 (1.10)	1.09 (1.12)	1.11 (1.15)	1.08 (1.12)	1.06 (1.08)	1.09 (1.12)
	Savings	1.06 (1.07)	1.07 (1.08)	1.12 (1.13)	1.07 (1.10)	1.10 (1.13)	1.14 (1.18)	1.03 (1.02)	1.06 (1.06)	1.08 (1.11)
	ACS4000	1.02 (1.01)	1.02 (1.03)	1.02 (1.02)	1.01 (1.00)	1.04 (1.06)	1.05 (1.06)	1.00 (1.00)	1.01 (1.02)	1.02 (1.04)
	ACS7000	1.02 (1.01)	1.02 (1.03)	1.01 (1.02)	1.00 (1.00)	1.04 (1.06)	1.04 (1.05)	1.00 (1.00)	1.01 (1.02)	1.02 (1.04)
Count Best Heuristic Sol. Identified	Greedy	0	0	0	1	0	1	0	0	0
	Sweep	0	0	0	0	0	0	0	1	0
	Savings	1	0	0	0	0	0	0	1	1
	ACS4000	8	11	10	5	6	5	4	5	4
	ACS7000	13	14	14	7	8	8	5	5	4
Avg. Sol. Time (Max Sol. Time)	Greedy	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.1)	0.0 (0.0)	0.0 (0.0)
	Sweep	0.0 (0.1)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.1)	0.0 (0.0)
	Savings	0.2 (0.5)	0.2 (0.5)	0.2 (0.5)	0.2 (0.4)	0.1 (0.4)	0.1 (0.3)	0.0 (0.1)	0.0 (0.1)	0.0 (0.1)
	ACS4000	9.9 (13.9)	9.6 (13.7)	8.9 (12.4)	9.6 (16.2)	8.5 (15.8)	7.8 (13.8)	10.1 (16.0)	9.3 (16.9)	6.8 (10.6)
	ACS7000	17.4 (24.2)	16.8 (24.0)	15.6 (21.7)	16.7 (28.2)	14.9 (27.6)	13.6 (24.1)	17.7 (28.1)	16.3 (29.7)	11.8 (18.7)

The first rows of results from Table 18 demonstrate that only the Greedy solution methods was able to identify feasible solutions for all of the tests instances while the

ACS procedure (at both values of M) was able to find feasible solutions for all but 1. The Savings algorithm was slightly worse as it was not able to identify feasible solutions in 6 of the 81 test instances while the Sweep algorithm performed the worst by not identifying feasible solutions in 12 of the 81 test instances. With respect to the Sweep heuristic, the inability to identify feasible solution is not surprising since it is the least ‘greedy’ of the different heuristics as it is solely based on network layout without the ability to fill nearly full routes with small capacity locations. Hence, the Sweep heuristic is not recommended to solve problems which have limited flexibility with respect to the combinations of stops which can be serviced by one vehicle.

With respect to the heuristic results compared with the exact solution method from Chapter 5, Table 18 indicates that the worst to best solution techniques are the Greedy, Sweep, Savings, and ACS solution methods for nearly all radius and test combinations. The only exceptions in this ordering are for the Augerat Set P test instances with radius cases B and C in which the Sweep method performed better on average than the Savings method.

The clear best solution method for these small test instances is the ACS as it finds a solution which is less than 4% greater on average compared with the exact solution method. This decreases to a gap of 2% or less on average if the radius case B and C instances from Augerat Set P are excluded. Furthermore, the ACS finds a better solution on average for the Christofides and Eilon radius case A test instances than the exact solution method from Chapter 5 (as some of these instances terminated at the 60 minute time limit). Table 18 reports these as even with a value of 1.00, but including more digits in the table would demonstrate that the ACS finds a solution value which is 99.87% of the exact solution method on average. Furthermore, when only the guaranteed optimal solutions are considered, as shown in parenthesis, the ACS finds the optimal solution in all of the Christofides and Eilon radius case A test instances.

With respect to the different radius cases, the heuristics generally perform better with a sparser coverage matrix. The possible cause of this performance is that the sparser matrix implies there are less feasible routes/solutions that can be generated. With respect to the different test instance sets, the results demonstrate that the heuristics find better solutions on average for the Christofides and Eilon instances as compared to both of the Augerat instances. The hypothesis for this difference is based on comparing Figure 10 which shows the solutions for the Christofides and Eilon 30 customer, 4 vehicle, and radius case A solutions with Figure 11 in subsection 6.3.3 which shows the solutions for the Augerat Set P 55 customer, 7 vehicle, and radius case A solutions. In general, the test instances from Augerat are more uniformly dispersed compared with the cases from Christofides and Eilon. Hence, it is hypothesized that the heuristic solutions are better on average for more clustered test instances compared with the more uniformly dispersed test instances. The justification for this hypothesis is that the developed heuristics place a significant emphasis on neighboring locations to create clusters (especially for the Savings and Sweep heuristics) which can benefit those instances in which natural clusters already exist.

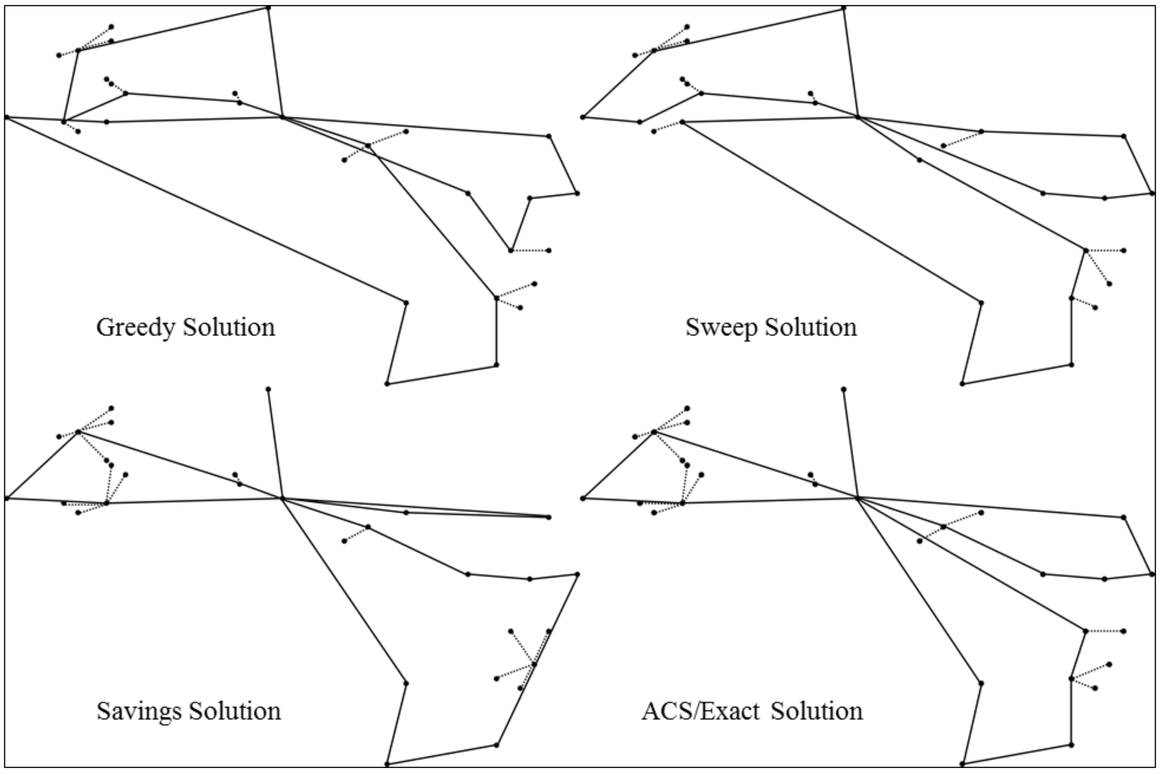


Figure 10. Heuristic and exact solutions for the Christofides and Eilon 30 location, 4 vehicle, and radius case A test instance.

Also demonstrated in Figure 10 is the advantages and disadvantages of the different solution techniques. For instance, the Greedy method clearly provides the weakest solution method as there are multiple long travel arcs and crossing vehicle routes. By comparison, the routes generated by the Sweep and Savings method are clearly superior to the Greedy routes even though they are not optimal. In addition, these two solutions share numerous similarities to the known optimal routes identified by the algorithm in Chapter 5 and the ACS heuristic. For instance, each of the solution methods identified that a vehicle always stops at one of the upper left most points in the network and at least serves three other coverable locations in its vicinity. It may be possible to use such information to further improve the solution methodologies by limiting the possible routing/covering combinations and focus the search techniques. However, further research is needed to investigate if such improvements are possible and whether or not they improve the solution quality.

The second to last rows of results in Table 18 demonstrate that the ACS solution method is able to find the best heuristic solution in all but 3 of the 81 test instances further implying that it is the best solution method for these small problems. The sole outliers are two test instances where the Savings method (2nd best solution method for small problem instances with respect to solution quality) identifies the best solution and one test instance where the Greedy method identifies the best solution. With respect to the latter, this situation only occurred because all of the other heuristics were unable to identify feasible solutions for this test instance. In addition, the differences in values between the ACS4000 and the ACS7000 show a subset of the total number of test instances which benefited from the larger value of M. Specifically, 58 out of the 78 test instances where ACS7000 identified the best solution already identified the same solution when $M = 4000$. The remaining 20 instances benefited from the longer running time.

The clear disadvantage of the ACS and the Savings methods is that the ACS methodology is the most computationally intensive while the Savings method is the second most computationally intensive as reported in the final rows of Table 18. While these times are still very small, they are significant in comparison to the Sweep method which was able to identify nearly as competitive of solutions in comparison to the Savings heuristic in a much shorter amount of time. Note that this advantage is potentially negated by the feasibility challenges of the Sweep methodology. Therefore, the ACS methodology is recommended for all small test instances unless computational time is of major concern. In such a case, either a smaller value for M can be employed (as was done for the ACS4000 results) or the Sweep/Savings methods can be used if algorithmic results are needed within a second.

6.3.3 Large test instances

The results shown in this section are for the test instances from Set A and Set P by Augerat and the test instances from Christofides and Eilon for any problems with 50 or more customers. The results from the 51 location, 5 vehicle Christofides and Eilon test instance and the 76 location, 4 vehicle Augerat's set P test instance are excluded to avoid overtuning. For each of the remaining test instances, three cases were created based on the radius options in Table 14.

The results from these tests are shown in Table 19 which has the same statistics as Table 18 with two exceptions. The first difference is that the second set of statistics in Table 19 calculate the average ratios of the heuristic solutions over the best solutions identified across all of the heuristics. This difference from Table 18 is required as these large test instances have yet to be solved optimally. The second difference is that the solution times shown in the last set of rows exclude the Augerat Set P test instance with 4 vehicles and 100 customers. This instance is excluded as its time results are a significant outlier which drastically alter the values shown in Table 19. The time results from this instance will be addressed in depth later in the section.

Table 19. Solution quality and solution time results from test instances with 50 or more customers (*excludes time results from Augerat Set P test instance with 4 vehicles and 100 customers)

Measure	Method	Augerat Set A			Augerat Set P			Christofides and Eilon		
		A	B	C	A	B	C	A	B	C
Test Instances	N/A	12	12	12	11	11	11	6	6	6
Count of Feasible Solution Identified	Greedy	12	12	12	10	10	10	6	6	6
	Sweep	6	6	6	6	6	6	4	4	4
	Savings	12	12	12	11	11	11	6	6	6
	ACS4000	12	12	12	10	10	10	6	6	5
	ACS7000	12	12	12	10	10	10	6	6	6
Ave. Heuristic Sol. over Best Heuristic Sol.	Greedy	1.21	1.25	1.23	1.18	1.20	1.25	1.26	1.26	1.32
	Sweep	1.13	1.16	1.16	1.06	1.09	1.11	1.05	1.09	1.11
	Savings	1.08	1.09	1.09	1.05	1.07	1.09	1.01	1.02	1.01
	ACS4000	1.00	1.00	1.00	1.01	1.00	1.01	1.03	1.03	1.01
	ACS7000	1.00	1.00	1.00	1.00	1.00	1.00	1.03	1.02	1.01
Count Best Heuristic Sol. Identified	Greedy	0	0	0	0	0	0	0	0	0
	Sweep	0	0	0	0	0	0	1	0	0
	Savings	0	0	0	2	2	1	2	4	3
	ACS4000	11	8	9	5	5	2	2	2	1
	ACS7000	12	12	12	9	9	10	4	2	3
Avg. Sol. Time (Max Sol. Time)*	Greedy	0.0 (0.1)	0.0 (0.0)	0.0 (0.0)	0.0 (0.2)	0.1 (0.6)	0.1 (0.6)	0.2 (0.6)	0.2 (0.7)	0.2 (0.6)
	Sweep	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.1 (0.2)	0.1 (0.4)	0.1 (0.4)	0.1 (0.4)	0.1 (0.4)	0.2 (0.5)
	Savings	2.0 (6.8)	2.0 (6.8)	2.0 (6.8)	1.1 (2.3)	1.1 (2.6)	1.1 (2.5)	5.1 (11.1)	5.1 (11.1)	5.2 (10.8)
	ACS4000	20.8 (34.9)	20.0 (35.4)	19.3 (35.5)	21.5 (61.9)	26.0 (110)	25.5 (113)	49.1 (102)	48.5 (107)	52.3 (105)
	ACS7000	36.5 (62.1)	35.2 (62.6)	33.3 (62.1)	39.4 (108)	47.7 (193)	47.5 (198)	85.7 (178)	85.0 (188)	82.2 (184)

The results from Table 19 continue to reinforce the observations from Table 18 that the Sweep heuristic is outperformed by all other heuristics as measured by the quantity of identified feasible solution. This is again theorized to be caused by the lack of a greedy mechanic in the Sweep algorithm. Unlike the results for the small test instance, the sole technique which was able to identify feasible solutions for all of the test instances was the Savings algorithm. Both the Greedy and ACS7000 solution methods were unable to find feasible solutions in 3 out of the 87 test instances while the ACS4000 solution method was unable to find feasible solutions in 4 out of the 87 test instances. Hence, by combining the results of Table 18 with those from Table 19, the Greedy and ACS algorithms perform the best with respect to finding feasible solutions while the Savings algorithm is second best.

The second set of results from Table 19 show that the Greedy heuristic is the worst heuristic regardless of the test instance set and the Sweep heuristic is the third worst heuristic. For nearly all of the test instances, the ACS procedure is the best performing heuristic while the Savings algorithm is the second best with the sole exception of the Christofides and Eilon radius case A test instances in which the Savings algorithm is able to outperform the ACS procedure by 2% on average. However, the ACS procedure is still recommended with respect to solution quality as it is able to outperform the Savings algorithm for all of the Augerat test instances by 5% on average in the worst case (Augerat Set P radius case A) and by as much as 9% on average in the best cases (a tie between three data set and radius combinations).

The split in results between the ACS procedure and the Savings procedure for the Augerat test instances compared with the Christofides and Eilon test instances are hypothesized to be caused by the differences in the test instances. For example, by comparing Figure 10 with Figure 11 (which shows the solutions obtained from the heuristics for the Augerat Set P 55 location, 7 vehicle, and radius case A test instance), the Christofides and Eilon test instances are more geographically clustered than the Augerat test instances. The effect of this clustering is that the techniques which place greater emphasis on the deterministic network location of two points (such as the Savings and Sweep method) are more likely to have success in such instances compared with the ACS technique which is a more stochastic methodology. This hypothesis is supported by Table 18 and Table 19 as the Sweep and Savings methods are much more competitive with the ACS procedure for the Christofides and Eilon instances compared with the Augerat instances. However, this effect is not large enough to recommend the Savings algorithm over the ACS procedure as the Savings algorithm outperforms the ACS procedure on average for *only* the Christofides and Eilon radius case A instances with 50 or more customers. In all other cases, the Savings algorithm performs as well as the ACS procedure on average or it performs significantly worse.

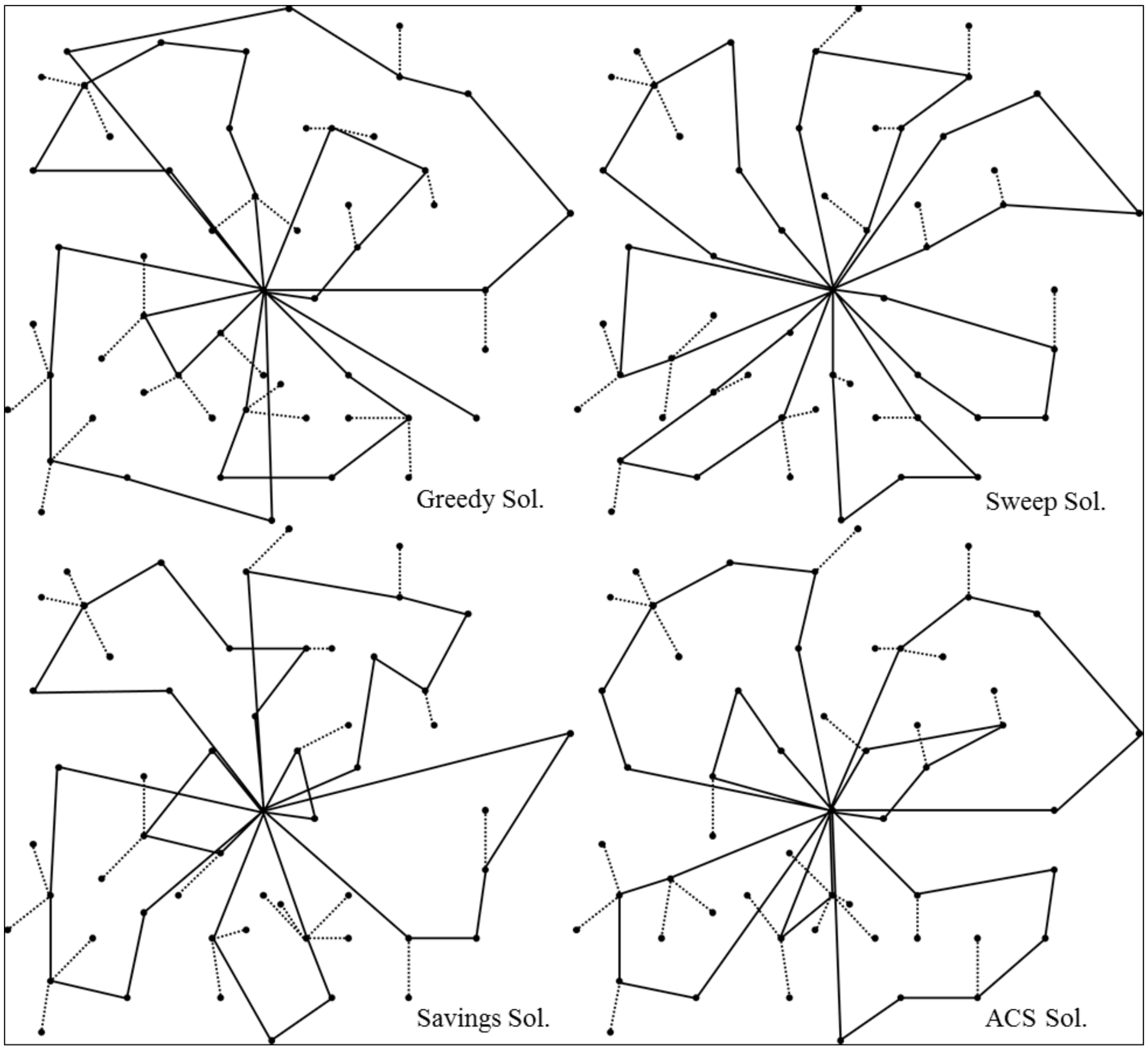


Figure 11. Heuristic solutions for the Augerat Set P 55 location, 7 vehicle, and radius case A test instance.

Recommending the ACS procedure is further validated by the count of test instances in which each solution heuristic identified the best solution. For instance, the ACS7000 procedure identified the best solution in 73 out of the 87 test instances (45 of these 73 instances had the best solution identified in the ACS4000 procedure as well) while the Savings procedure identified the best solution in 14 of the 87 instances. This demonstrates the dominance of the ACS procedure in these tests. Furthermore, even in the Christofides and Eilon radius case A instances where the Savings procedure outperformed the ACS procedure with respect to average solution quality, the ACS procedure identified 4 of the best solutions out of 6 of the test instances while the Savings procedure was able to identify the best solution in the other 2 instances. This demonstrates that the average case results discussed previously are driven by a sole outlier (specifically the Christofides and Eilon radius case A instance with 76 locations and 14 vehicles) where the ACS procedure terminated with a solution 16% higher than the Savings algorithm.

The final results shown in Table 19, the average solution times, again demonstrate that the ordering of the heuristics from least to most time intensive are the Greedy, Sweep, Savings, and ACS heuristics. In general, the solution times for all but the ACS heuristic are minimal. For the ACS7000 heuristic, the solution times are drastically larger with the maximum solution times of 3.3 minutes while the ACS4000 heuristic had a maximum time of 1.9 minutes. Note that the Augerat Set P test instance with 4 vehicles and 100 customers was omitted from these time results since its ACS7000 average solution time was 13 hours (ACS4000 average solution time was 8 hours), the average solution times for the Savings heuristic was 1.9 minutes, and the average solution times for the Sweep heuristic was 2.6 minutes. These poor results are caused by the large number of locations in combination with the low number of vehicles. Hence, the branch-and-bound tree investigated in COVROUTE is significantly larger for this problem than in any of the other test instances. These results demonstrate that larger instances will need to modify the route construction operation if competitive solution times are desired.

6.4 Discussion

Four heuristic solution methods were presented in this chapter (Greedy, Sweep, Savings, and ACS heuristics) and extensive computational tests were performed on all adapted test instances from Set A and Set P by Augerat and all test instances by Christofides and Eilon. The results demonstrated that the ACS heuristic is preferred for all instances with the sole exception of large problems where the network is not geographically dispersed and the coverage matrix is sparse as the Savings technique is able to perform better on average for these cases. However, the poor average case performance in this scenario is solely motivated by one test instance whose results are an outlier when compared to the rest of the instances. After the ACS procedure, the Savings heuristic is the second best technique for nearly all cases. With respect to solution time, the ACS algorithm was the most computationally complex while the Savings heuristic provided the second worst results but most instances still terminated within seconds. For practitioners where a balance between solution quality and time is required, the ACS procedure is still recommended, but with a smaller value of M such as 4000 or less as the results from these experiments identified the same solution as the experiments when $M = 7000$ in 68% of the test instances.

Future research is recommended on numerous aspects of the presented algorithms. The most important recommendation is to investigate different route improvement/construction techniques. In these implementations, COVROUTE was developed to improve routes using a branch-and-bound methodology. COVROUTE was selected as it is a simplified version of the column generation procedure $NEVAL$ used in the exact algorithm from Chapter 5. While it demonstrates that it can generate high quality covering routes, it was the principal reason why running time started to increase as the number of vehicles remained low and the number of customers increased. Hence, a simpler process should be investigated if larger CCVRP instances are to be solved or if other metaheuristics are to be tested such as Tabu Search or Simulated Annealing which may require more uses of a route construction/improvement technique. Similarly, a more complex version of COVROUTE can be beneficial for improving the solution quality of the smaller results without significant impact on overall solution time. For instance, the 2-opt heuristic used to create feasible cycles could be replaced with more advanced techniques such as an optimal route solver or the 3/4-opt heuristics.

An additional improvement is to apply common advances for the adapted heuristics to their CCVRP implementations. For instance, the Greedy algorithm can potentially be improved in two methods. The first is to improve how covered locations are assigned to stops. As opposed to an index-based method, it may be possible to solve a knapsack problem or consider which locations are not coverable from other locations and give priority to covering such locations. Additionally, a distance threshold can be applied such that the Greedy heuristic does not travel further than a given distance unless necessary. This will stop the heuristic from greedily filling the end of the routes with small capacity locations and creating the long arcs observed in Figure 10 and Figure 11. The Sweep and Savings heuristics also have numerous advancements which could be applied to the CCVRP heuristics. Practitioners interested in such advancements are referred to Laporte and Semet (2014) who summarize a majority of the recent modifications to these algorithms. Finally, numerous ACS improvements are possible including using a multi-ant colony approach (Bell and McMullen 2004) or incorporating additional heuristic components such as a scatter search methodology (Zhang and Tang 2009) or genetic mutations (Bin, Zhong-Zhen, and Baozhen 2009). Finally, it may also be possible to adapt other common CVRP heuristics to the CCVRP. For instance, numerous metaheuristics have been applied to the CVRP including the Genetic Algorithm, Tabu Search, and Simulated Annealing. Those interested in such techniques are referred to Gendreau, Laporte, and Potvin (2001) who summarizes these additional techniques and their applications to the CVRP. Research is recommended to see how these techniques can be adapted to the CCVRP and to test their solution quality against the heuristics presented in this article.

6.5 Conclusion

With respect to the mobile food retailer routing and scheduling problem, any of the developed heuristics can be employed by a mobile food retailer as none of the solution procedures require the use of commercial software. In fact, each procedure only needs the specification of the customer network and a few details regarding the vehicles such as the vehicle’s capacity. With respect to choosing between the different approaches, the recommendations are the same as those for the general practitioners. In general, the ACS algorithm is recommended except in the case where there are a large number of stops per route as the Savings algorithm provides competitive results in considerably less time.

In summary, the advantage of the heuristics developed in this chapter is that they scale well with respect to the customer network assuming that the number of stops per route does not become too large. Therefore these heuristics are a more robust route building heuristic, compared with the exact algorithm from Chapter 5, which can assist mobile food retailers with improving their economic sustainability by increasing their revenues while decreasing their transportation costs. Furthermore, the ACS algorithm is particularly well suited for mobile food retailers as it can incorporate nuances which may exist in particular mobile retailer implementations. For instance, it can easily build a set of routes which do not serve all customers in the situation the retailer does not have the requisite capacity and/or vehicles or the ACS can be adapted to add additional constraints such as limiting the number of stops at certain types of service locations. Making such modifications require mobile food retailers to partner with the appropriate technical staff and practitioners, but the ability to generate custom routes for a specific retailer outweigh these challenges.

CHAPTER 7

OPERATIONAL MOBILE FOOD RETAILER DECISIONS: A CASE STUDY

The four prior chapters of this dissertation detailed multiple operational tools to address the mobile food retailer product mix problem and the mobile food retailer routing and scheduling problem. The purpose of this chapter is to discuss a case study using collected operational data to demonstrate how these tools can be used effectively by an actual mobile healthy food retailer as well as demonstrate that these tools can lead to a more successful intervention methodology by increasing the economic sustainability of the mobile retailer. In particular, two approaches will be utilized for determining the optimal product mix. The first will utilize the two constraint version of the problem (i.e. the DKP) while the second will feature more constraints (i.e. the MDMKP) to ensure that multiple substitutable products are excluded. Following these problems, the routing and scheduling of the retailer will be addressed based on the particular requirements of the case study retailer. To properly address this application, an additional solution methodology will be presented which is able to incorporate multiple time windows for each possible stop. The results from these tests with respect to both the specific mobile food retailer which serves as the basis of this case study and general mobile retailers will be discussed.

7.1 Phoenix mobile fresh food retailer

The mobile retailer which serves as the basis of this case study is called Fresh Express and operates in the metropolitan Phoenix area. Specifically, Fresh Express' main area of operation is the Discovery Triangle region of Phoenix (shown in Figure 12). The

Discovery Triangle region is a classic food desert as many of the residents in these neighborhoods are low income and there are only 7 full service supermarkets in the area (and only 2 full service supermarkets if Tempe is excluded). Fresh Express was started in April 2014 and in 2015 has completed almost 7,500 transactions which include selling nearly 80,000 units of fresh fruits and vegetables (Discovery Triangle 2016).

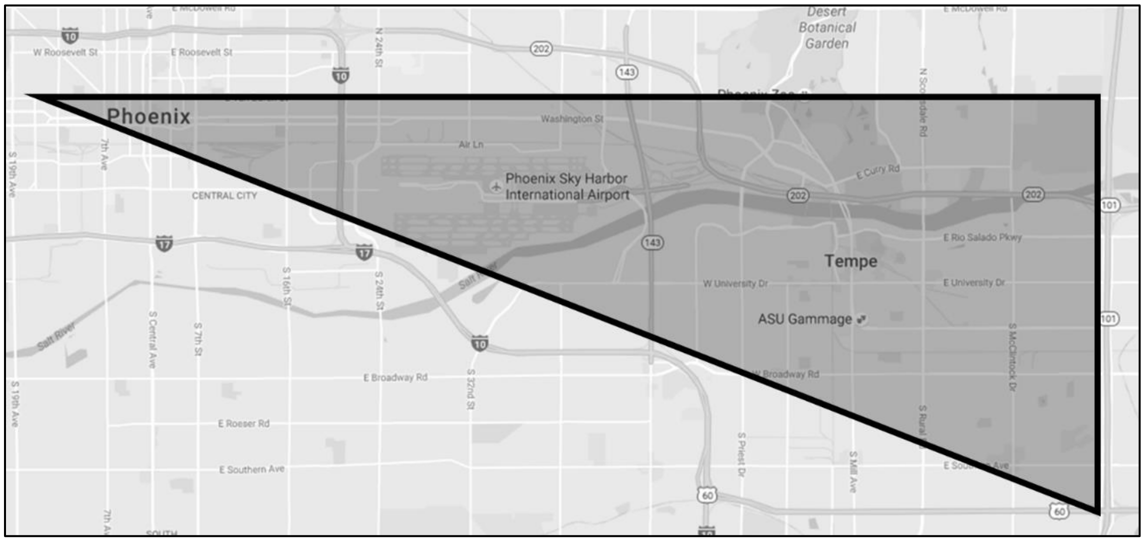


Figure 12. Discovery Triangle region of Phoenix, AZ

Operationally, Fresh Express uses a single retrofitted City of Phoenix bus which has all interior seats and structures removed. In their place are removable racks which are stocked with up to 45 bins. These bins hold a variety of fresh fruits and vegetables which are stocked at or below the prices offered by supermarkets. Currently, Fresh Express' only physical installation is a large walk-in cooler which it stores on the grounds of the not-for-profit corporation UMOM. Fresh Express parks its bus at a City of Phoenix transit facility and its driver is typically a City of Phoenix bus driver whose employment is subsidized by the city. The only other permanent employees associated with the retailer are an Executive Director who manages all operational decisions and two to three retail employees who travel with the bus and handle all transactions and manage the bus while in transit.

Fresh Express currently has two suppliers who exclusively provide the fresh fruits and vegetables which are stocked on the retailer. The first and principal supplier is Peddler's Son which is a local Arizona wholesaler of fresh fruits and vegetables. The secondary supplier is a local grower who has just started to supply locally grown produce when it is in season. The produce stocked on Fresh Express includes seasonally stocked goods such as pumpkins and sweet potatoes, locally-desired goods such as jalapenos and avocados, and staple good such as apples and lettuce. For the convenience of customers, these items are typically priced at values rounded to the nearest quarter. In addition, SNAP purchases are also accepted on Fresh Express which research has shown is crucial for food desert residents. Furthermore, Fresh Express typically offers matching funds for SNAP purchases where every one dollar spent using SNAP dollars qualifies for purchasing two dollars worth of food.

Fresh Express has over 35 potential service locations within the Phoenix area. The most common types of stops are elementary schools, older adult housing communities, and community centers. However, Fresh Express also stops at locations which are not necessarily targeted at low-income populations such as downtown Phoenix and the Arizona State University downtown campus. These stops serve as marketing opportunities to increase the visibility of the retailer which is crucial to ensure continued sponsor support. Some of the aforementioned stops also have specific time periods when Fresh Express wishes to stop. For instance, elementary schools are typically only visited at the start or end of the school day to coincide with the times when parents would likely be visiting the schools.

Fresh Express typically has the time and resources to visit four stops per route with the exception of routes which service downtown Phoenix which only service three stop per route. Each stop is typically for one hour with some exceptions such as downtown Phoenix which is a two hour stop. As of September 2016, Fresh Express has three routes scheduled per week in the Discovery Triangle region but some weeks have an additional route to test the service of Fresh Express in the south and west Phoenix area. These routes are scheduled a month at a time and posted online. Some stops are visited every week while some are only visited once a month. This rate of service is typically based on the level of demand at each stop so that those with more demand are visited more frequently.

In addition to these details which influence the case study, there are other key details which impact the success of Fresh Express. While these are not the focus of this dissertation, they should be acknowledged as any successful mobile food retailer must include similar techniques. For instance, Fresh Express stocks recipes which utilize the ingredients they sell. This not only increases the appeal of such items on the retailer, but it also teaches low-income consumers how items can be prepared as many low-income shoppers frequently cite they do not know how to prepare less common produce items. Similarly, Fresh Express has partnered with a local chef to prepare cooking demonstrations which serve the same purpose as the recipes. Finally, Fresh Express requires any organization which serves as a service location to advertise that Fresh Express will be stopping at their location thereby increasing the visibility, popularity, and revenue of Fresh Express.

7.2 Designing a simple product mix

The first component of this case study is the development of a product mix for Fresh Express which can be modeled as a simple DKP. For this problem, one of the key constraints is that the space of the retailer cannot be exceeded. The other constraint is motivated by the crucial issue for Fresh Express: the inability to meet a given profit margin. Currently, Fresh Express is only able to cover the cost of its produce. This is clearly not a sustainable strategy, especially if Fresh Express were to lose sponsorships or grants which cover other operational and labor costs. Hence, a demand constraint is added to the model which requires the stocked mix to meet a stated profit threshold.

In addition, Fresh Express would ideally like to address two issues with this product mix. The first is that the mix should be as healthy as possible while the second is that the mix should be as low cost as possible. Hence, the model for this section of the case study is to

Maximize: $\sum_{i \in I} h_i x_i,$ (7-1)

Minimize: $\sum_{i \in I} c_i x_i,$` (7-2)

subject to

$\sum_{i \in I} v_i x_i \leq V,$ (7-3)

$\sum_{i \in I} p_i x_i \geq P,$ (7-4)

$x_i \in \{0,1\} \quad \forall\, i \in I.$ (7-5)

In this multi-objective model, the set I represents the complete set of possible produce items which can be stocked and x_i represents the binary decision to include or exclude an item from being stocked in Fresh Express. For each item $i \in I$, h_i represents the ‘healthiness’ of item i, c_i represents the customer cost of item i, v_i represents the amount of space each item occupies on the Fresh Express, and p_i represents the profit earned from selling an average bin of item i. Finally, V represents the space available on Fresh Express while P represents the required profit from the stocked product mix.

To develop set I, approximately three months of data was collected from Fresh Express including the total number of units sold for each produce item. The 44 items which were stocked with the most regularity were selected for inclusion in set I as it was assumed the data for these items would be less influenced by outliers. To add more potential items to I, a local supermarket chain (specifically Bashas) provided a year of produce shipment data from its warehouses to each of its stores. From this data, an additional 51 items were added to I which represent the items which were most frequently shipped by Bashas which are not currently stocked with regularity by Fresh Express.

For each $i \in I$, it is assumed $v_i = 1$ as this would represent Fresh Express stocking one bin of food with that item. Since Fresh Express can stock 45 such bins, it was assumed $V = 45$. To determine c_i for all $i \in I$, two methods were employed. If the item is stocked by Fresh Express, c_i was set to the price of one serving of the item as sold Fresh Express and if the item is not stocked by Fresh Express, the value of c_i was set to the non-sale price of one serving of that item as sold by Bashas. A similar methodology was used to calculate p_i for all $i \in I$. If an item i was sold by Fresh Express, the average profit per week is used as p_i. If an item i is not sold by Fresh Express, then Bashas data was used to determine the expected profits/week if the item were to be sold by Fresh Express. This was completed by comparing the overlapping foods (i.e. those sold by both

Fresh Express and by Bashas) and then determining a scaling ratio to apply to other 51 food items.

To determine the healthiness scores of each item (i.e. h_i), a nutritional profiling system, such as the commercially available NUVAL system, was initially considered. However, these systems will normally score fresh fruits and vegetables at the highest level which allows for little differentiation for the items in the current study. Instead, nutritional data per serving was acquired for each of the food items. The nutritional data points were selected such that they match those utilized by the USDA to create the Thrifty Food Plan (Carlson et al. 2007). These categories include protein, thiamin, riboflavin, niacin, folate, calcium, phosphorus, potassium, magnesium, iron, zinc, fiber, cholesterol, sodium, and fat as well as vitamins A, B6, B12, C, and E. Each food item's nutritional data was normalized and summed to form one score. Any nutrient which has a positive health benefit (e.g. calcium) had a positive influence on the score and any nutrient which has a negative health benefit (e.g. fat) had a negative influence.

The remaining key issue is that the DKP model which best describes the simple mobile retailer product mix decision for Fresh Express is multi-objective as demonstrated by (7-1) and (7-2). Since the solution methodology for the DKP only has a single objective, a convex combination of (7-1) and (7-2) was calculated such that

$$\text{Maximize: } \lambda \sum_{i \in I} h_i x_i - (1 - \lambda) \sum_{i \in I} c_i x_i$$

was the new objective for the DKP. By varying λ to a fine degree, the efficient frontier for the optimal product mix was approximated. Based on the opinions of Fresh Express in regards to the acceptable balance between health and customer cost, the implemented product mix can be selected from this frontier.

In total, values of P were tested from \$150 to \$300 in increments of \$25. The lowest value was selected as the next lowest increment, \$125, provided nearly the same solutions across all values of λ compared to $P = \$150$. These varying values for P were conducted to demonstrate the effect of profit requirements on the identified solution. For each level of P, λ was varied between 0 and 1 and each instance was solved using DKPSOLVE from Chapter 4. The graphical representation of these frontiers is shown in Figure 13 where each labeled line represents a different profit requirement. The vertical axis represents the consumer cost if a customer were to purchase one of each item while the horizontal axis is the sum of the health scores for each selected item. Specifically, each line corresponds to a different setting for P and every point where two line segments meet represent a unique product mix. The product mixes along the line which are further to the right side of Figure 13 correspond to solutions when λ approached one (i.e. the healthiness of the mix was the dominant concern) while those to the left side correspond to solutions when λ approached zero (i.e. the consumer cost of the mix was the dominant concern).

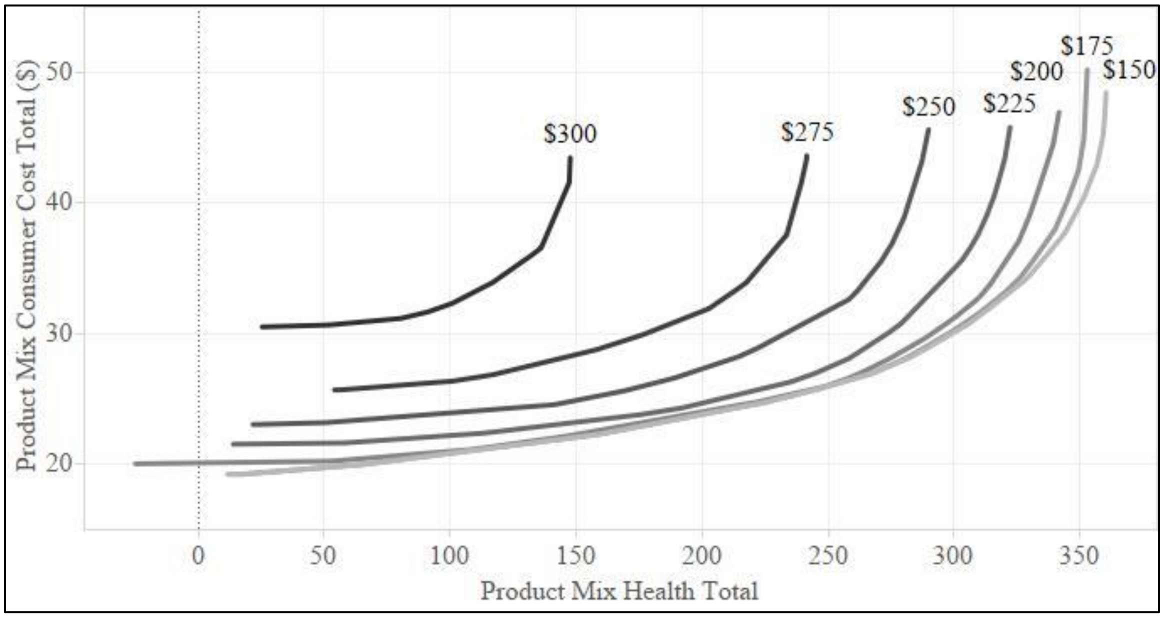


Figure 13. Product mix efficient frontiers for simple Fresh Express product mix

To better demonstrate the meaning of the cumulative health scores, consider a hypothetical product mix which has a cumulative health score of 250. Since Fresh Express stocks 45 bins of food items, this is equivalent to stocking items who have an average health score of 5.56. This value implies that the average grocery item stocked in this product mix has a cumulative nutritional value which is 5.56 standard deviations above the average. One possible example of this number is that this hypothetical average item has a 1.00 standard deviations better value for protein, thiamin, riboflavin, niacin, and folate, a 0.56 standard deviations better value for iron, and all of the other 14 nutrients are at the average value calculated across all of the potential 95 items. One clear flaw of this approach is that the stocked product mix may be deficient in one or more nutrients. Since the mobile retailer only represents a subset of the grocery purchases for a consumer (as many staple items such as bread and milk are not stocked), such potential nutritional deficits are not expected to have significant health impacts which would require a more nuanced approach to building the product mix.

The results shown in Figure 13 demonstrate that the variety in feasible product mixes for Fresh Express depends upon the required profit margin. For instance, when the required profit margin is minimal (such as $150 or $175), the lines nearly overlap. This demonstrates that Fresh Express can increase its required profit from $150 without significantly effecting either the health or consumer cost of its product mix. In addition, both of these profit margin requirements result in frontiers which span a majority of Figure 13 therefore implying that there is a high variety of possible product mixes if the retailer maintains a low profit margin requirements. Similarly, the $200 profit margin frontier overlaps with only the middle sections of the $150 and $175 frontiers. That indicates that if Fresh Express desires a balance between consumer cost and health, a profit margin of $200 results in competitive mixes (with respect to the multiple objectives) compared with mixes which have required profit margins of $150 or $175. However, as one of the objective becomes the dominant focus, the $200 mix either becomes less healthy or less cost friendly compared with the product mixes at $150 or $175 as demonstrated by the growing distance between the three lines towards the left and right sides of the figure respectively.

For the remaining required profit margins, the lines no longer overlap which indicates that requiring profit margins of $225 or above starts to have an impact on the healthiness and consumer cost of the mix regardless of the balance between the competing objectives. Initially, the impact of this growing requirement is minimal as demonstrated by the distance between the $200 and $225 frontiers, but this distance grows between each increment in P. For example, the identified optimal plan when P was \$300 and $\lambda = 0$ (i.e. the healthiness of the product mix is inconsequential) had a consumer cost of \$30.51 which represents how much it would cost to buy one unit of every item on Fresh Express. This is does not compare favorably to the product mixes when the required profit margin was \$275, \$250, and \$225 which had consumer costs of \$25.69, \$23.04, and \$21.54 respectively. In fact, the cumulative consumer cost when the required profit margin was \$150 was \$19.24. Hence the increase in consumer costs between profit margin levels of \$275 and \$300 is almost as large as the consumer cost increase between profit margin levels of \$150 and \$275. Therefore, the mix which requires the highest margin is not advisable as this greatly increases the costs payed by all consumers. A similar trend occurs when $\lambda = 1$ (i.e. the consumer cost of the product mix is inconsequential) as the healthiness score decreases from 360.4 when the required profit margin is \$150 to healthiness scores of 341.7, 289.9, and 147.7 when the required profit margin is \$200, \$250, and \$300 respectively.

By using this approach, Fresh Express can select the product mix which best meets the modeled objectives based on their desired tradeoffs between profit margin, consumer cost, and health or they can select between the developed mixes based on any non-modeled criteria. Currently, Fresh Express' weekly profits are in the lower range of the values tested. The specific values are discussed in the next subsection. This not only provides validation of the data, but also demonstrates that Fresh Express may be able to become economically stronger if they were to alter their product mix.

It should also be noted that this methodology *can* return practical product mixes. For example, when $P = \$250$, the product list shown in Table 20 is along the efficient frontier. This mix's total health offering is 287.4 (average over all optimal offerings in this frontier is 270.9) and the total cost to customers is \$43.26 (average over all optimal offerings in this frontier is \$40.69). This mix is particularly practical as it does not feature many substitutable products (i.e. products where if one item is not offered, then the other product would be considered a viable substitute) and it represents an even balance between health and consumer costs. Hence, the employed formulation, even though it is relatively simple, is useful without adding additional constraints which could complicate simple solution methodologies. This is important since mobile retailers would not have access to sophisticated software packages.

Table 20. Sample simple Fresh Express product mix for $P = \$250$

Gala Apples	Broccoli Crowns	Coconut	Sunflower Seeds	Yukon Gold Potatoes
Plantains	Green Cabbage	Kiwi	Green Onion	Spinach Salad Mix
Bananas	Regular Carrots	Papaya	Navel Oranges	Butternut Squash
Green Beans	Bi-Color Corn	Artichoke	Valencia Oranges	Grey Squash
Pinto Beans	White Corn	Asparagus	Gold Bell Pepper	Yellow Squash
Snap Peas	Cucumber	Bean Sprouts	Red Bell Pepper	
Blackberries	Red Grapes	Brussel Sprouts	Anaheim Chili	
Raspberries	Collard Greens	White Mushrooms	Red Plums	
Strawberries	Iceberg Lettuce	Grapefruit	Russet Potatoes	
Broccoli Florets	Kale	Lemon	Sweet Potatoes	

However, Table 20 and the methodology used to select the mix represented in the table does highlight one of the flaws in using a DKP to model the mobile food retailer product mix problem. As shown in Table 20, there are three (out of a possible four) types of berries stocked on the retailer in this plan and both types of corn are also included in the stocked mix. In fact, the product mix shown in Table 20 is one of the mixes which feature the least quantity of substitutable items compared with all of the other identified product mixes. The next section within this chapter discusses an approach to address this issue.

7.3 Designing a complex product mix

One of the major issues with the prior approach is that some of the identified product mixes were not practical. For instance, some of the mixes along the approximated efficient frontiers in Figure 13 included multiple copies of substitutable goods such as two types of corn and all four types of possible berries. While stocking these goods is not an issue in traditional supermarkets, the smaller space of mobile retailers requires a more efficient use of available resources. To avoid these types of issues, additional constraints were added to limit the number of substitutable goods. This new model is

Maximize: $\lambda \sum_{i \in I} h_i x_i - (1-\lambda) \sum_{i \in I} c_i x_i$ (7-6)

subject to

$\sum_{i \in I} v_i x_i \leq V,$ (7-7)

$\sum_{i \in I} p_i x_i \geq P,$ (7-8)

$\sum_{i \in I} b_{iJ} x_i \leq B_J \quad \forall J \in \mathcal{J}$ (7-9)

$x_i \in \{0,1\} \quad \forall\, i \in I.$ (7-10)

In this updated approach, which is now an MDMKP, the set $\mathcal{J}$ represents a partitioning of I into pairwise mutually exclusive subsets. Each $J \in \mathcal{J}$ represents a set of items which consumers would consider as substitutable. For instance, one J contains three elements which represent three different varieties of apples (specifically gala, golden delicious, and granny smith) and it is assumed that a customer would be willing to buy a gala apple if a granny smith were not offered. A summary of the elements of $\mathcal{J}$ and the number of elements represented by each J are shown in Table 21.

Table 21. Partitioning and counts of possible produce items for Fresh Express

Apple (3)	Collard Green (1)	Kale (1)	Pineapple (1)
Artichoke (1)	Corn (2)	Kiwi (1)	Plantain (1)
Asparagus (1)	Crouton (1)	Leaf Lettuce (4)	Plum (1)
Avocado (1)	Cucumber (1)	Leek (1)	Potato (3)
Banana (1)	Eggplant (1)	Lemon (1)	Radish (1)
Beet (1)	Garlic (1)	Lime (1)	Rice (1)
Berries (4)	Grapefruit (1)	Mango (1)	Spaghetti Squash (1)
Broccoli (2)	Grapes (2)	Melon (3)	Sprout (e.g. Alfalfa) (2)
Brussel Sprout (1)	Green Bean (2)	Mushroom (1)	Summer Squash (3)
Butternut Squash (1)	Green Onion (1)	Pinto Bean etc. (2)	Sunflower Seed (1)
Cabbage (2)	Guacamole (1)	Onion (3)	Sweet Pepper (3)
Carrot (2)	Herb (4)	Orange (4)	Sweet Potato (1)
Cauliflower (1)	Hot Pepper (4)	Papaya (1)	Tomatillo (1)
Celery (1)	Jicama (1)	Peach (1)	Tomato (2)
Coconut (1)	Juice (1)	Pear (1)	

For each $J \in \mathcal{J}$, $b_{iJ} = 1$ if item $i \in J$ and $b_{ij} = 0$ otherwise. The value for B_J depends on the number of product items in set J. If there is either one or two items in J, then $B_J = 1$ (i.e. the retailer can stock at most one of the substitutable items), otherwise $B_J = 2$ (i.e. the retailer can stock at most two of the substitutable items). Hence, constraint set (7-9) will ensure that the product mix features a wide variety of food similar to the offerings found in traditional grocery stores.

The computational tests for the presented MDMKP model were conducted using the same approach as the DKP model. Specifically, values between 0 and 1 were incrementally tested for λ to approximate the efficient frontier and values for P were tested between \$150 and \$275 in increments of \$25. This latter range differs slightly from the approach used for the DKP model as the MDMKP has no feasible solutions when $P = \$300$. These problem instances were solved using the MDMKP Fixed-Core procedure using the largest core size δ_G tested from Chapter 3. The Fixed-Core procedure was selected due to the simplicity of the implementation and the simplicity of the test instances (95 variables) which permitted larger core sizes without a large effect on solution times. The approximated efficient frontiers from these test instances are shown in Figure 14 which follow the same conventions as Figure 13.

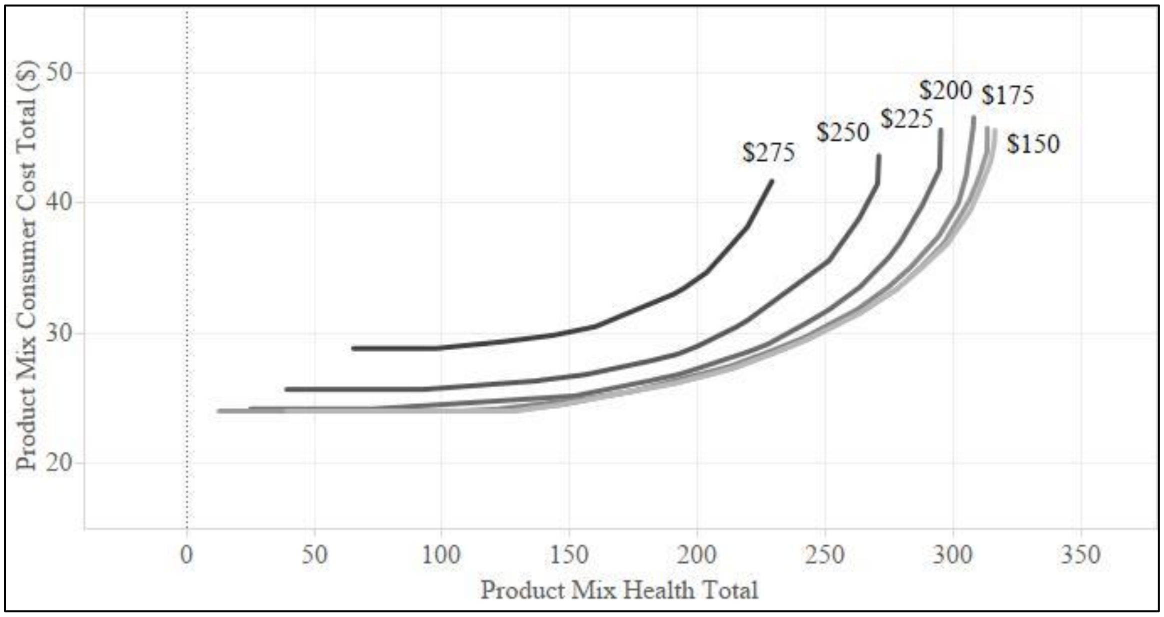


Figure 14. Product mix efficient frontiers for complex Fresh Express product mix

Figure 14 demonstrates that there is a minimal loss in the healthiness of the mix or the consumer cost of the mix if the mobile retailer required \$150, \$175, or \$200 in profit from the stocked mix as these lines overlap for a majority of their frontiers. Hence, the retailer can safely stock a more profitable product mix (up to \$200) without significantly impacting the health or consumer cost of the stocked product mix. This trend even continues for the \$225 profit product mix if the healthiness of the product mix is deemphasized as the \$225 frontier overlaps with the previously mentioned frontiers towards the left side Figure 14. Finally, there is only a sizeable effect on both the health and consumer cost of the product mix when a profit margin exceeding \$250 is required. This is especially significant for mixes with a profit margin exceeding \$275 as these mixes have noticeably higher consumer costs and lower healthiness ratings across all values of λ.

The advantage of including constraints (7-9) is that all of the identified product mixes represent consumer friendly solutions. For instance, Table 22 shows the updated solution when $P = \$250$ and λ equals the same value used to generate the plan in Table 20. While the initial plan shown in Table 20 had a low number of substitutable goods, the upgraded plan in Table 22 is superior as at most two of any substitutable items are now recommended for being stocked on the retailer.

Table 22. Sample complex Fresh Express product mix for $P = \$250$

Gala apples	Whole Carrots	Artichoke	Valencia Oranges	Butternut Squash
Plantains	White Corn	Asparagus	Gold Peppers	Grey Squash
Bananas	Cucumbers	Bean Sprouts	Red Peppers	Tomatillos
Green Beans	Red Grapes	Brussel Sprouts	Green Chilis	Yellow Squash
Pinto Beans	Collard Greens	White Mushrooms	Jalapeños	Spaghetti Squash
Blackberries	Iceberg Lettuce	Grapefruit	Plums	
Raspberries	Kale	Lemon	Russet Potatoes	
Cauliflower	Coconut	Sunflower Seeds	Yukon Gold Potatoes	
Broccoli	Kiwi	Green Onion	Sweet Potatoes	
Cabbage	Papaya	Navel Oranges	Spinach Salad Mix	

The clear disadvantage of this approach is that adding restrictions to the product mix model results in solutions which feature lower objective values. For instance, the plan in Table 22 has a healthiness score of 270.6 and a consumer cost of $41.46 (combined objective value of 92.7) which are together worse in comparison to the less restricted plan shown in Table 20 which had a healthiness score of 277.4 and a consumer cost of $43.26 (combined objective value of 98.9).

To better demonstrate the effects of constraint set (7-9) on the solution performance, a graphical summary of the differences between a subset of the frontiers from Figure 13 and Figure 14 is shown in Figure 15. In particular three profit requirements are shown in Figure 15 ($150, $200, and $250) and the frontiers from both Figure 13 and Figure 14 are included. For the two smaller values of P, there is a large difference between solving the

DKP and MDMKP versions of this problem which indicates that the product mixes identified for the DKP with small values of P are likely to feature a large number of substitutable goods and are therefore more impractical. However, the frontier gap decreases between the DKP and MDMKP solutions when $P = \$250$ therefore implying that the solutions obtained for this profit value for the DKP (such as the solution Table 20) are more likely to be practical stocking options.

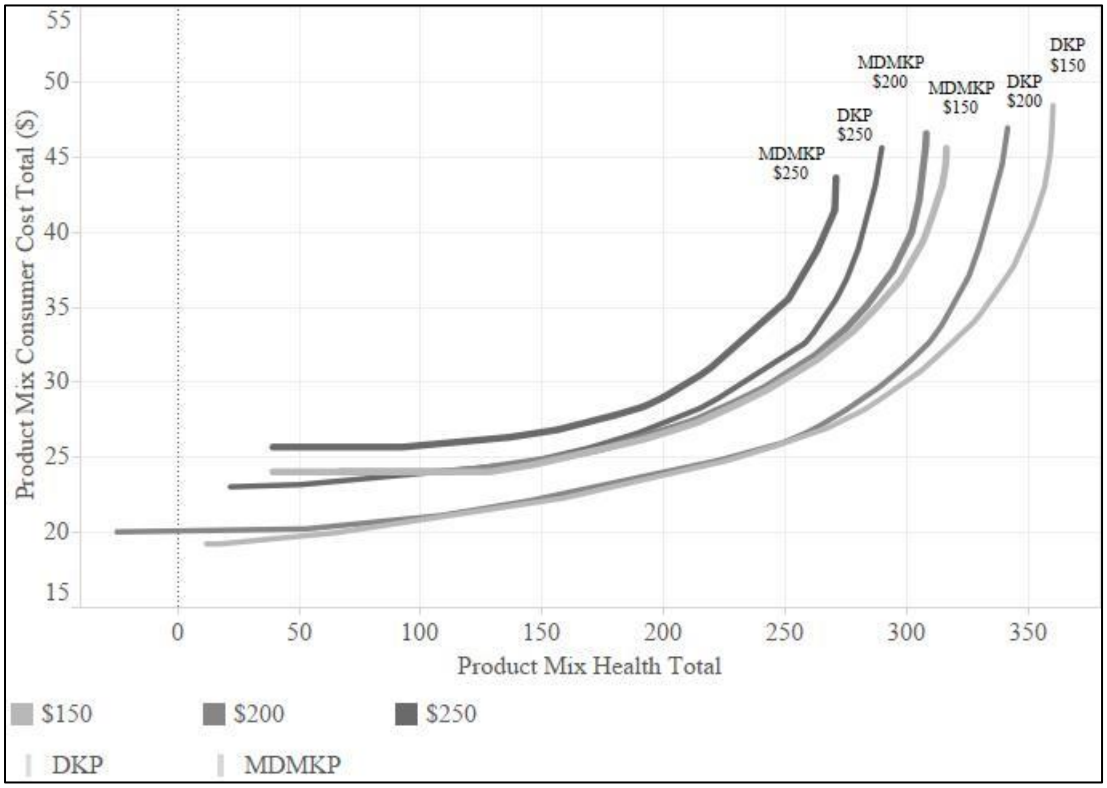


Figure 15. Efficient frontier differences between the simple and complex Fresh Express product mixes

To determine how these solutions (both in subsection 7.2 and subsection 7.3) compare with Fresh Express, the 45 items which were stocked the most frequently on Fresh Express was determined from the operational data. Based on the same coefficients used in the DKP and MDMKP in this case study, the cumulative consumer cost of these items is $41.01, the healthiness score of this mix is 16.3, and the profit of this mix is $174.53. With respect to the DKP portion of this case study, this product mix is far off even the $175 efficient frontier. For instance, at the same profit margin and healthiness score, Fresh Express could lower the consumer cost of its mix to at least $19.24. For the same profit margin and consumer cost, Fresh Express could increase the nutritional score of their mix to 345.2. Finally, for the same healthiness score and consumer cost, Fresh Express could increase their profit margin to $275. In fact, a mix on the $275 efficient frontier has a better healthiness score, 233.6, and consumer cost, $37.56, than Fresh Express' current product mix.

However, these DKP solutions may not be practical as they feature substitutable product. Therefore, the current Fresh Express product mix is also compared to the MDMKP efficient frontiers. For the same profit margin and healthiness score, Fresh Express could lower the consumer cost of its mix to at least $24.04. For the same profit margin and consumer cost, Fresh Express could increase the nutritional score of their mix to 306.7. Finally, for the same healthiness score and consumer cost, Fresh Express could increase their profit margin to $275. In fact, a mix on the $275 efficient frontier has a better healthiness score, 219.6, and consumer cost, $38.16, than Fresh Express' current product mix. Hence, Fresh Express can increase their daily profits by $100 (an increase of over 150% of the current profits) even in the most conservative estimates which will greatly increase their chances of achieving economic sustainability.

In summary, Fresh Express can use either methodology to develop a product mix, but the MDMKP model is recommended as it develops mixes which feature greater item variety which more closely resembles the mix of grocery items found in traditional supermarkets. An additional advantage of using the MDMKP approach is that it is possible to add additional constraints to the model based on the needs of Fresh Express or similar retailers. The clear disadvantage is that only the poorest performing solution method for MDMKPs from Chapter 3 can solve the problem without commercial software. For smaller problems, this is not expected to significantly affect solution quality or time, but interested mobile retailers are recommended to pursue partnerships with practitioners to gain access to higher quality solution methods.

Within the solutions obtained from the MDMKP model, selecting the final product mix is at the discretion of Fresh Express. However, it is recommended that Fresh Express select the minimum profit margin which would satisfy its needs and then determine an adequate balance between health and consumer cost based on the approximate frontiers displayed in Figure 14. Using this methodology, it is nearly guaranteed that Fresh Express would not be able to identify a better product mix meeting this criterion. Fresh Express is also welcome to select a product mix which it prefers based on some criteria which was not included in the model. For instance, all of the developed MDMKP product mixes could be provided which meet a certain profit margin and Fresh Express could select the mix which includes the most staple items for that region. This consideration could either be included in further iterations of the model as a constraint/objective or this consideration could continue to be used as a decision making criteria to select between different generated solutions. Regardless of their selected mix, the case study demonstrated that Fresh Express is able to increase the profits of their product mix by nearly $100/day with no impact to consumer costs or health thereby greatly increasing their chances at achieving economic sustainability.

7.4 Fresh Express routing and scheduling

Given a product mix, the remaining issue is to design a feasible routing and scheduling plan for Fresh Express. As discussed at the start of this chapter, Fresh Express operates in the Discovery Triangle region of Phoenix but they are currently testing routes in the south Phoenix and west Phoenix areas. These areas are similar to the Discovery Triangle region as they have a large number of low-income residents and full service supermarkets are sparse. The purpose for testing these routes is that Fresh Express is contemplating adding full time routes in these areas either on additional days of the week with their current bus or adding a second bus which exclusively serves these regions. To assist in this work, Fresh Express would like advice on designing these routes. The purpose of this section is to demonstrate how this issue can be addressed.

In particular, the routing and scheduling model which was designed specifically for Fresh Express will be discussed first. A new model is needed, which expands on the models developed in Chapters 5 and 6, as Fresh Express requires specific time windows for service at its stops. Since the prior models do not include such considerations, a new model was added. This model is introduced in this chapter, as opposed to having its own chapter, as theoretical experiments and extensive computational results have yet to be performed on the model. Following this model, the data used in the model is described prior to discussing the results and recommendations for Fresh Express.

7.4.1 Fresh Express routing model

The CCVRP model presented in Chapter 5 serves as the basis of the model for Fresh Express. However, Fresh Express' location have strict time window requirements which must be met for service. Specifically, each location typically has two possible time windows when service can occur. Every elementary school service location has time windows associated with the start and end of the school day while every other location has two time windows which correspond to the morning period and afternoon period. These latter two windows are needed as Fresh Express does not operate between Noon and 1 PM with the sole exception of the downtown Phoenix stop, but this stop will not be included in this portion of the case study as it is not geographically in the study zone.

To formulate the model for a generic mobile food retailer scheduling model with multiple time windows, the following sets are required: J which is the set of vehicles, M which is the set of demand locations and the vehicle depot (which is indexed by 0), and W_m which represents the set of time windows for location $m \in M$. For the decision variables, let $y_{mnj} = 1$ signify that vehicle $j \in J$ travels from location $m \in M$ to $n \in M$ and let $y_{mnj} = 0$ otherwise. Also let $\omega_{mnji} = 1$ signify that location $m \in M$ is served during time window $i \in W_m$ by vehicle $j \in J$ stopping at location $n \in M$ and $\omega_{mnji} = 0$ otherwise. Finally, let d_{mj} represent the time when vehicle $j \in J$ starts service at location $m \in M$ and let w_{mj} represent the waiting time of vehicle $j \in J$ at location $m \in M$ which includes the time to service m and any other applicable locations covered by a vehicle at m.

Given these sets and variables, the Fresh Express model is to

Maximize: $\sum_{m\in M}\sum_{n\in M}\sum_{j\in J}\sum_{i\in W_m} p_m\omega_{mnji} - \sum_{m\in M}\sum_{\substack{n\in M\\ n\neq m}} \tau_{mn}y_{mnj}$ (7-11)

subject to

$\sum_{\substack{m\in M\\ m\neq n}} y_{mnj} - \sum_{\substack{o\in M\\ o\neq n}} y_{noj} = 0 \quad \forall\, n\in M, j\in J,$ (7-12)

$\sum_{n\in M\setminus\{0\}} y_{0nj} = 1 \quad \forall j\in J,$ (7-13)

$t^{max}\left(1-\sum_{m\in M} y_{mnj}\right) \leq d_{nj} \quad \forall n\in M, j\in J,$ (7-14)

$w_{mj} \geq I_{mj} \quad \forall m\in M, j\in J,$ (7-15)

$d_{mj} + w_{mj} + tt_{mn} - d_{nj} \leq \left(1-y_{mnj}\right)N \quad \forall m\in M\setminus\{0\}, n\in M, j\in J,$ (7-16)

$d_{mj} \leq t^{max} \quad \forall m\in M, j\in J,$ (7-17)

$d_{mj} \geq tt_{0m} \quad \forall m\in M, j\in J,$ (7-18)

$\sum_{m\in M} y_{mnj} \geq \sum_{i\in W_n} \omega_{nnji} \quad \forall n\in M, j\in J,$ (7-19)

$\omega_{mnji} \leq b_{mn}\left[1-\frac{1}{t^{max}}\left(d_{nj}-l_{mi}\right)\right] \quad \forall m\in M, n\in M, j\in J, i\in W_m,$ (7-20)

$\omega_{mnji} \leq b_{mn}\left[1-\frac{1}{t^{max}}\left(e_{mi}-\left[d_{nj}+w_{nj}\right]\right)\right] \quad \forall m\in M, n\in M, j\in J, i\in W_m,$ (7-21)

$\sum_{n\in M}\sum_{j\in J}\sum_{i\in W_m} \omega_{mnji} \leq 1 \quad \forall m\in M,$ (7-22)

$\sum_{m\in M}\sum_{n\in M}\sum_{i\in W_m} u_m\omega_{mnji} \leq C_j \quad \forall j\in J,$ (7-23)

$d_{mj} \geq 0 \quad \forall m\in M, j\in J,$ (7-24)

$y_{mnj}, \omega_{mnji} \in \mathbb{B} \quad \forall m\in M, n\in M, j\in J, i\in W_m.$ (7-25)

The goal of this model is to maximize the profit of the routing plan which is a function of the profit (p_m) earned from serving each location $m\in M$ less the cost of traveling (τ_{mn}) between every pair of locations $m\in M$ and $n\in M$. This is constrained by 14 sets of constraints. Constraint set (7-12) ensures that cycles are constructed in the service network while constraint set (7-13) ensures that each cycle contains the depot. Constraint set (7-14) sets all d_{nj} variables to exceed t^{max} (the time limit indicating when the vehicle must be returned to the depot) if a location $n \in M$ is not serviced by a vehicle while constraint set (7-15) sets the waiting time at location $m \in M$ to at least exceed the required time to service that location (I_{mj}) for vehicle $j \in J$.

Constraint set (7-16) ensures that if a vehicle $j \in J$ travels between locations $m \in M$ and $n \in N$ that the arrival at location n is at least the sum of the arrival at location m plus the waiting/servicing time at location m and the travel time between locations m and n (tt_{mn}). N is a large constant. Constraint sets (7-17) and (7-18) ensure that d_{mj} does not exceed t^{max} (in combination with (7-14) this will ensure any location not serviced by vehicle j will have $d_{mj} = t^{max}$) and is at least greater than the travel time required to travel from the depot to location m. Constraint set (7-19) ensures that if a vehicle stops at a given location, it must service the demand at that location during one of its time windows. This constraint can be excluded from other applications, but it is included for Fresh Express as it is assumed that customers would be unsatisfied being served by a vehicle at a distance if another vehicle were to actually stop at their location but not service the customers at that location.

Constraint sets (7-20) and (7-21) are the most complicated constraints and ensure the covering logic is maintained in the developed solution. To better demonstrate the function of these constraints, a subset of an example network is provided in Figure 16. In this figure, the seven points represent possible demand points which are indexed as 2 through 8. The points which are shaded as yellow represent points which are physically visited by the mobile retailer. These points are referred to as “visited” locations. Hence, the retailer visits demand points 3, 5, and 7 in order as indicated by the set of arrows. The points shaded green represent locations within the service radius of a visited location which are represented by the red circles. These points are referred to as “serviced” locations. In Figure 16, demand points 2, 4, 5, 6, and 8 are potential serviced locations as they are within the service radius of visited point 5.

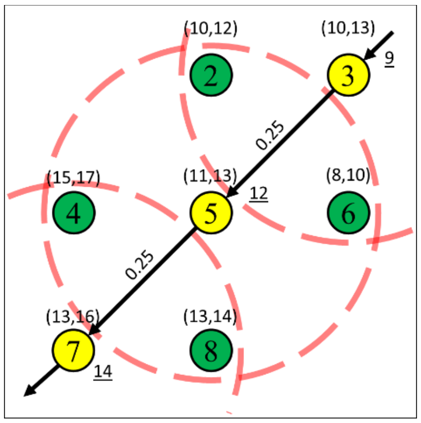


Figure 16. Sample service network with time windows.

Also provided in Figure 16 are the time windows for each location provided in parentheses above each circle. The first number is the start of the time window while the second number is the end of the time window. For this demonstration of the constraints, it is assumed that each stop only has one time window. If the retailer starts service within this window, the demand at that location can be captured. Also provided in the figure are the travel times between each visited location (listed along the arcs) as well as the time of arrival at each visited location given as the underlined number and it is assumed that it takes 0.50 hours to service each possible location.

To better explain the figure, service location 5 will be used as an example. The mobile retailer started servicing location 5 at 1200 and since it took 0.25 hours to travel between serviced locations 3 and 5, the retailer stayed at location 3 for 2.75 hours. Using this same logic, the retailer stayed at location 5 for 1.75 hours prior to traveling to location 7. A summary of the schedule is provided in Table 23 which shows the schedule for the retailer (shown by the diagonal lines) along with the service time windows (shown by the shading).

Table 23. Time windows and service windows for sample service network and routing plan.

Location	**Time of Day**								
	0800	**0900**	**1000**	**1100**	**1200**	**1300**	**1400**	**1500**	**1600**
2			▒	▒					
3		▨	▨▒	▨▒	▒				
4								▒	▒
5				▒	▨▒	▨			
6	▒	▒							
7						▒	▨▒	▨▒	▨
8						▒			

Based on Table 23 and Figure 16, the vehicle is at location 5 from 1200 to 1345 in which time it can possibly service locations 2, 4, 5, 6, and 8 (as indicated by the red circle around location 5 in Figure 16) so long as the visit is within the appropriate time windows for each location. Since the time windows for locations 2, 5 and 8 coincide with the time when the retailer is visiting location 5, locations 2, 5 and 8 can have their demand satisfied. Since location 6 has a time window which ends prior to location 5 being visited, its demand cannot be serviced. Likewise, location 4's time window starts after the retailer leaves location 5, so its demand cannot be serviced.

Constraint sets (7-20) and (7-21) are designed to determine if serviced location m's time windows (green circles in Figure 16) are satisfied by a visit to a demand location n over a specific time interval. Specifically, constraints (7-20) determine if the start of a visit to location n is after serviced location m's service window $i \in W_m$, signified by l_{mi}, while constraints (7-21) determine whether the end of a visit to location n is before serviced location m's service window $i \in W_m$, signified by e_{mi}. Both of these constraints feature three key elements. The first is the "normalizing constant" which is the fractional value $(t^{max})^{-1}$ in both constraint sets. Secondly, each have a "time window calculation" which is expressed as $(d_{nj} - l_{mi})$ and $(e_{mi} - [d_{nj} + w_{nj}])$ in constraints (7-20) and (7-21) respectively. Finally, both have the binary parameter b_{mn} which indicates if location m can be serviced from a vehicle stopping at location n.

To demonstrate these constraints, consider the visit to location 5 and the service of location 8. Location 5 is visited from 1200 until 1345 while the time window of location 8 is 1300 to 1400. Hence, location 8's demand can be satisfied and it should be possible for $\omega_{85j1} = 1$ as this would indicate location 8 can be serviced from location 5. For some vehicle j, a constraint (7-20) is

$$\omega_{85j1} \leq b_{85}\left[1 - \frac{1}{t^{max}}\left(d_{5j} - l_{81}\right)\right]$$

for visiting location 5 and servicing location 8. Note that $b_{85} = 1$ as this covering is possible so if $\omega_{85j1} = 1$ is possible, then $\frac{1}{t^{max}}\left(d_{5j} - l_{81}\right) \leq 0$. Since $d_{5j} = 12$ (start of visit to location 5) and $l_{81} = 14$ (last possible start of service for location 8), then the time window calculation is strictly negative. In the general case, this time window calculation will be non-positive so long as $d_{nj} \leq l_{mi}$ (i.e. the start of a visit occurs before the end of the service time window). Therefore,

$$\frac{1}{t^{max}}\left(d_{5j} - l_{81}\right) < 0$$

which implies ω_{85j1} is not restricted to be 0.

Likewise, a constraint (7-21) is

$$\omega_{85jd} \leq b_{85}\left[1 - \frac{1}{t^{max}}\left(e_{81} - \left[d_{5j} + w_{5j}\right]\right)\right]$$

for visiting location 5 and servicing location 8. Note that $b_{85} = 1$ as this covering is feasible so if $\omega_{85j1} = 1$ is possible, then $\frac{1}{t^{max}}\left(e_{81} - \left[d_{5j} + w_{5j}\right]\right) \leq 0$. Additionally observe that w_{5j} (the time spent at location 5 by vehicle j) is bounded below by I_{5j} from constraint set (7-15) and is bounded above by $w_{5j} \leq d_{7j} - d_{5j} - tt_{57}$ from constraint set (7-16) (note that $y_{57j} = 1$ based on the routing from Figure 16). Since $d_{7j} = 14$, $d_{5j} = 12$, and $tt_{57} = 0.25$, then $0 \leq I_{5j} \leq w_{5j} \leq 1.75$. Given this bound and that $d_{5j} = 12$ and $e_{81} = 13$, then $e_{81} - \left[d_{5j} + w_{5j}\right]$ is at most $1 - I_{5j} = 0.5$ and is -0.75 at a minimum. This implies the term can be non-positive if necessary. Therefore, it is possible for $\frac{1}{t^{max}}\left(e_{81} - \left[d_{5j} + w_{5j}\right]\right) \leq 0$ thereby permitting ω_{85j1} to either be 0 or 1.

In conclusion, when the time windows of a serviced location overlap with a visited location, as was demonstrated with locations 8 and 5 respectively, neither constraints (3-26) or (3-27) restrict ω_{mnji} to be strictly less than 1. Hence, it is possible to satisfy the demand at location m from location n.

Consider the opposite case such as servicing location 6 from visited location 5 in Figure 16. Since the time window of location 6 completely occurs before location 5 is visited, as demonstrated in Table 23, the demand at location 6 should not be captured for vehicle j. To demonstrate this logic, consider constraint (7-20) which is

$$\omega_{65ji} \leq b_{65}\left[1 - \frac{1}{t^{max}}\left(d_{5j} - l_{61}\right)\right]$$

for visiting location 5 and servicing location 6. Note that $b_{65} = 1$ as this covering is possible so if $\omega_{85j1} = 1$ is forbidden, then $\frac{1}{t^{max}}\left(d_{5j} - l_{61}\right) > 0$. Since $d_{5j} = 12$ and $l_{61} = 10$, then the time window calculation is strictly positive. Therefore, $\frac{1}{t^{max}}\left(d_{5j} - l_{61}\right)$ is strictly positive thereby indicating location 6 cannot be serviced from location 5 by vehicle j. Additionally, the normalizing constant $(t^{max})^{-1}$ will ensure $\frac{1}{t^{max}}\left(d_{nj} - l_{mi}\right) \leq 1$ in these cases. Hence, ω_{65j1} will never be restricted from being 0 by constraints (7-20) which is necessary given that it is a binary variable.

To demonstrate another case, consider servicing location 4 from visited location 5 in Figure 16. Since the time window of location 4 completely occurs after location 5 is visited, as demonstrated in Table 23, the demand at location 4 should not be captured for vehicle j. To demonstrate this logic, consider constraint (7-21) which is

$$\omega_{45j1} \leq b_{45}\left[1 - \frac{1}{t^{max}}\left(e_{41} - \left[d_{5j} + w_{5j}\right]\right)\right]$$

for visiting location 5 and servicing location 4. Note that $b_{45} = 1$ as this covering is possible so if $\omega_{45j1} = 1$ is forbidden, then $\frac{1}{t^{max}}\left(e_{41} - \left[d_{5j} + w_{5j}\right]\right) > 0$. As was demonstrated previously, w_{5j} is bounded such that $I_{5j} \leq w_{5j} \leq 1.75$. Since, $e_{41} = 15$ and $d_{5j} = 12$, then the time window calculation is at most 3-I_{5j} $= 2.5$ and is 1.25 at a minimum. Hence, $\frac{1}{t^{max}}\left(e_{41} - \left[d_{5j} + w_{5j}\right]\right)$ is strictly positive thereby indicating location 4 cannot be serviced from location 5 by vehicle j. Additionally, the normalizing constant

$(t^{max})^{-1}$ will ensure $\frac{1}{t^{max}}\left(e_{mi} - [d_{nj} + w_{nj}]\right) \leq 1$ in these cases. Therefore, ω_{45j1} will never be restricted from being 0 by constraints (7-21) which is necessary given that it is a binary variable.

The final major constraints sets are (7-22) and (7-23) which ensure that at most one vehicle services a location and that the capacity of vehicle $j \in J$ (C_j) is not exceeded based on the cumulative sum of served demand at locations $m \in M$ (u_m), respectively. The remaining two constraints ensure all d_{mj} are nonnegative and that the remaining variable sets are binary.

7.4.2 Fresh Express routing model solution algorithm

With the inclusion of time windows in this model, the prior solution models developed for the generic mobile retailers in Chapters 5 and 6 are no longer applicable. Hence, a new solution methodology was developed specifically for solving the Fresh Express routing model and similar models with time window constraints. This solution methodology is discussed as part of the case study, as opposed to having its own dedicated methodological discussion, because full algorithmic testing and experimentation has yet to be performed. The lack of these tests, which is recognized as a shortcoming at this time, are discussed in Chapter 8.

To solve this time window variant of the CCVRP specifically developed for Fresh Express, a Tabu Search heuristic was developed similar to the heuristic from Cordeau, Laporte, and Mercier (2001). The Tabu Search developed by Cordeau, Laporte, and Mercier was initially designed as a unified heuristic that could solve a variety of CVRP variants including those with time windows. Specifically, Cordeau, Laporte, and Mercier developed the approach for the periodic and the multi-depot CVRP with time windows. This approach was selected due to its flexibility to incorporate all of the nuances present in the Fresh Express CCVRP problem. The heuristic's simplicity is an additional advantage as it can be readily adapted to develop routes for CCVRPs with nuances different than those detailed for Fresh Express.

Prior to outlining the full details for the Tabu Search used to solve the Fresh Express routing model, the general heuristic details will be provided which apply to traditional CVRPs assuming vehicles cannot satisfy demand from a distance and a service location has a single time windows. These assumptions, and all others which do not apply to Fresh Express, will be revisited once the general Tabu Search heuristic is outlined.

During every phase of the Tabu Search procedure, a solution $s \in S$ represents a set of $|J|$ routes such that every customer belongs to exactly one route. Note that this solution may violate various constraints such as the cumulative load serviced by the vehicle, the time windows for service at a stop, or the total time of travel for the vehicle. Therefore, for any $s \in S$, let $q(s)$ represent the total load violation of the routes (servicing more demand than possible across all vehicles), let $d(s)$ represent the duration violation of the routes (cumulative time returning to the depot after t^{max} for all vehicles), and let $w(s)$ represent the time window violation of the routes. The calculation for $w(s)$ is $\sum_{j \in J} \sum_{m \in M^j} (d_n - l_n)^+$ where $M^j \subseteq M$ is the subset of location serviced by vehicle j.

Each solution s is then scored according to a function $f(s) = p(s) - \alpha q(s) - \beta d(s) - \gamma w(s)$ where $p(s)$ is the profit earned by solution s which is a function of the reward for demand serviced minus the cost of the routing. In $f(s)$, α, β, and γ are all positive parameters which are dynamically adjusted based on the current solutions identified within the Tabu Search. These parameters therefore control how the algorithm traverses the solution space.

Within the Tabu Search procedure, each solution s is defined based on an attribute set $B(s) = \{(m, j)\}$ which indicates location m is serviced by vehicle j in solution s. The Tabu Search proceeds by exploring the direct neighborhood of s (i.e. $N(s)$) by removing some $(m, j) \in B(s)$ and replacing it with attribute (m, j') where $j \neq j'$. Customer m is inserted in the route of vehicle j' by placing it in between the two consecutive stops which maximizes the value of $f(s)$ and route j is reconnected by having j travel directly from the predecessor of customer m to the successor of customer m. When this occurs, the attribute (m, j) is made Tabu for θ iterations which forbids customer m from being added back to vehicle j for a given duration. It is possible to revoke this Tabu status in the case where adding m to j would result in a better feasible solution than the best solution containing that attribute thus far in the algorithm. This logic is commonly referred to as the short-term memory of the Tabu Search heuristic.

The long-term memory of the Tabu Search heuristic tracks the frequency of an attribute in a solution s to further increase the diversity of the identified solutions. Specifically, if $f(s') < f(s)$ for some solution $s' \in N(s)$, then $f(s')$ is penalized in proportion to the long-term memory multiplied by a scaling factor. To define this long-term memory, let p_{mj} store the number of times attribute (m, j) has been added during the Tabu Search procedure. Hence, if $f(s') < f(s)$ then subtract $d(s') = \lambda p(s')\sqrt{|M| * |J|}p_{mj}$ from $f(s')$. This additional term penalizes $f(s')$ in proportion to the size of the customer network and vehicle set as well as in proportion to the solution reward. Also included in this penalty is a scaling parameter $\lambda > 0$ which can be used to control the intensity of the penalty. If $f(s') \geq f(s)$, then let $d(s') = 0$. The impact of this penalty is that the algorithm explores unvisited areas of the solution space when a local optima is identified

Given these definitions, the full Tabu Search procedure can be detailed below assuming that s^* stores the best feasible solution identified at that point in the heuristic:

CVRP Tabu Search Heuristic

```
Initialize first solution $s$ and $\alpha, \beta, \gamma$, and $\lambda$.
*If* $s$ is feasible, *then* let $s^* = s$ and $p(s^*) = p(s)$, *else* let $p(s^*) = -\infty$.
*For* $\kappa = 1, \ldots, \eta$, *do*
    Select $s' \in N(s)$ which maximizes $f(s') + d(s')$ that is not Tabu (i.e. the
        added attribute has been added in the past $\theta$ iterations) unless $s'$ is the best
        feasible solution identified thus far with that attribute.
    *If* $s'$ is feasible and $p(s') < p(s^*)$, *then* $p(s^*) = p(s')$ and $s^* = s'$.
    Update $\alpha$, $\beta$, and $\gamma$.
    Let $s = s'$.
Improve $s^*$ if possible.
```

To determine the initial s in the Tabu Search heuristic, numerous methodologies are possible. Within this implementation, a specific methodology is used based on the nuances of Fresh Express which will be detailed later. Readers interested in a more general approach for determining s are recommended to refer to Cordeau, Laporte, and Mercier (2001)j. Furthermore, α, β, and γ are initialized to 1 at the start of the algorithm (but this can be modified at the discretion of the practitioner) and they are updated throughout the procedure by a factor of $1 + \delta$ where $\delta > 0$. Specifically, α is multiplied by $1 + \delta$ if $q(s') > 0$ and α is divided by $1 + \delta$ if $q(s') = 0$, β is multiplied by $1 + \delta$ if $d(s') > 0$ and β is divided by $1 + \delta$ if $d(s') = 0$, and γ is multiplied by $1 + \delta$ if $w(s') > 0$ and γ is divided by $1 + \delta$ if $w(s') = 0$.

This procedure is sufficient for the general CVRP, but not for the Fresh Express model. First and foremost, not all customers are expected to be served by at once. To address this difference, the number of routes created for each $s \in S$ represent $|J| + 1$ routes where the last route is a 'dummy' route. This dummy route does not factor into the calculation of $q(s)$, $d(s)$, and $w(s)$ and serves as storage for all of the unserved customers. The second major modification is to incorporate the 'covering' mechanic by modifying the rules on how $N(s)$ is explored. Specifically, whenever a customer m is removed from j, it is first checked to see if it can be covered by any customer in each route $j' \neq j$ and if so, $f(s')$ is calculated. m is then tested for insertion and direct visitation into each j' and $f(s')$ is again calculated for each possibility. The insertion which results in the highest value for $f(s')$, either directly being visited or being served from a distance, is retained as one possible option for $s' \in N(s)$. In addition, a customer m which is directly visited and serves as a central site for serving other nearby customers in s cannot be a removed attribute in this step of the Tabu Search heuristic. Instead, all customers which are served at a distance by vehicle j from location m must first be removed from vehicle j prior to removing service location m. Other, more complicated, methodologies could be employed in this scenario, but this approach was selected due to its ease of interpretation and implementation.

7.4.3 Fresh Express routing data

Given this routing model and solution methodology, data was collected and aggregated regarding potential customers in the west and south Phoenix neighborhoods. According to the service area defined by Fresh Express, the south Phoenix community is bounded by Buckeye Rd., Baseline Rd., 24th St., and 35th Ave. while the west Phoenix community is two separate zones with the more southerly zone bounded by Buckeye Rd., Camelback Rd., Central Ave., and 67th Ave. and the northerly zone bounded by Camelback Rd., Northern Ave., Central Ave., and 43rd Ave. These regions are shown in Figure 17. The first step in estimating the demand for Fresh Express in these areas was to identify all possible service locations within these communities. To complete this process, aerial photography was utilized to identify all possible stopping locations by scanning each block within the service area. Any structure with a parking lot with adequate space for a bus to stop was noted and specific attention was given to elementary schools, day cares, college campuses, vocational schools, community centers, parks, and older adult living facilities as these are the traditional types of service locations for Fresh Express. In total, this resulted in 71 total elementary or similar schools, 14 housing complexes, and 18 other types of locations.

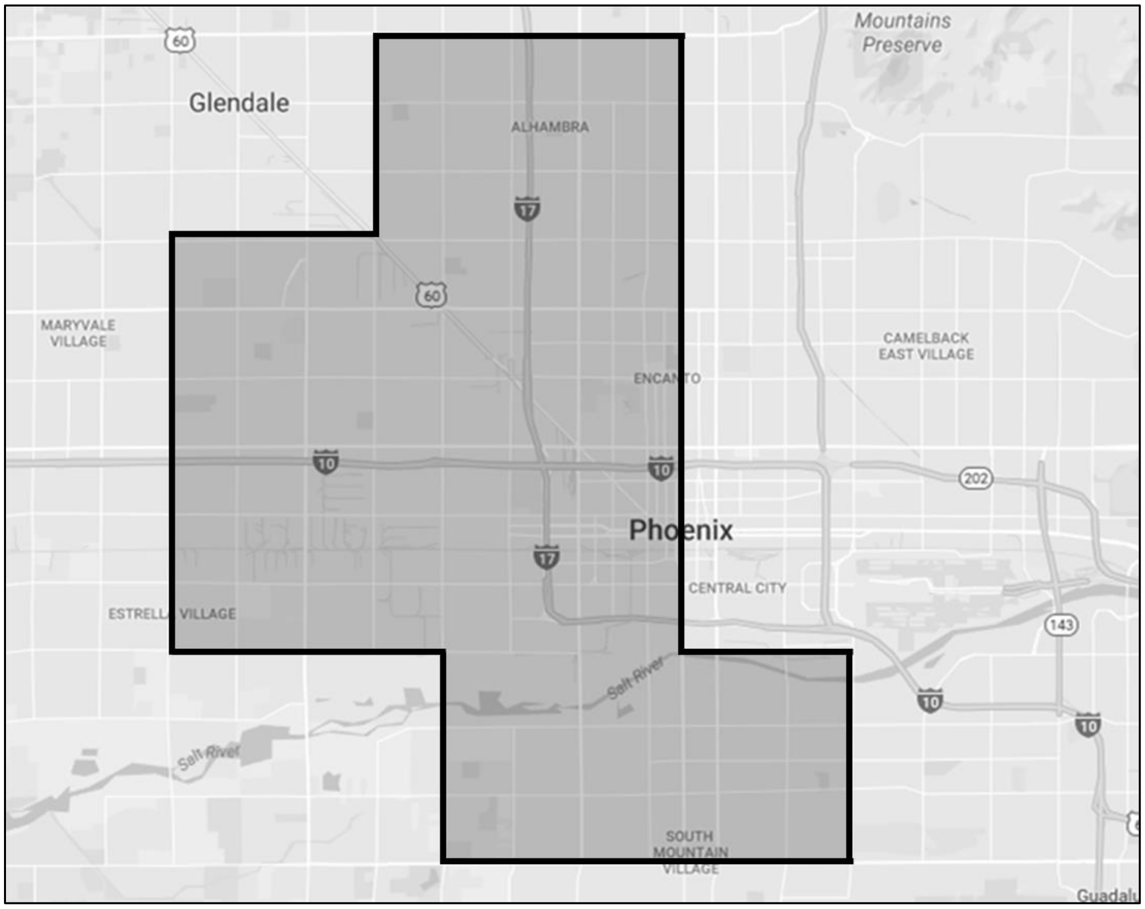


Figure 17. West and south Phoenix operational area for Fresh Express

Given these possible locations, data was collected regarding each stop in order to help estimate demand at each location. This data uses Fresh Express' current product mix as it is assumed all locations have the same demand profile for grocery items. The specific data collected depended on the type of stop. For instance, for every public school, data was obtained about the number of students in kindergarten through 4th grade, the number of students on free or reduced lunch, and the number of students who are English as a Second Language (ESL). If the school was not publicly listed, the school was either directly contacted or visited to obtain enrollment figures. For any applicable housing centers (typically low-income or older adult), the number of units were obtained either through listings or direct verification. For all other types of stops, general usage numbers were obtained through direct contact/interviews such as the number of visitors to a community center at peak times, etc. In addition, the distance to the nearest full-service supermarket was recorded for every potential stop.

To estimate demand at these locations, four months of operational data (March – June) were obtained for 2016 from Fresh Express. Matching data was then collected for each of these stops such as the size of the school or housing complex. In total, there are 41 observations of Fresh Express serving housing complexes, 17 observations of Fresh Express serving schools or day cares, and 86 observations of Fresh Express serving other types of stops. Based on data records, distributions were fit to the average revenue earned for every housing unit, for every child, and at every non-housing/non-school stop. In addition, stops which are currently in the south or west Phoenix areas (as Fresh Express is trialing some routes in these communities) had separate distributions fit to their specific data if enough data points existed for the service location. Given these distributions, it was assumed that south and west Phoenix would have experience the same revenue per child (in the case of schools), revenue per unit (in the case of housing complex), and revenue per site (in the case of all other locations without their own fitted distribution) as the locations already serviced by Fresh Express.

With these distributions, a stochastic optimization approach was conducted to identify a high-quality routing plan for Fresh Express. Specifically, 31 total test scenarios were created by sampling a variety of cases from each of the 103 potential service locations in the west and south Phoenix communities. The first test scenario represented the average from each distribution for each service location. For each of the next 25 test scenarios, a set of 50 samples were generated for each service location and the average revenue for each possible customer location was calculated across these 50 samples. This sampling and averaging procedure was repeated 25 times to obtain the 25 test scenarios. This procedure was motivated by two primary concerns: the possible sampled revenues for each site had high variability and it was assumed Fresh Express' primary concern was maximizing their expected earnings. Hence, generating 50 samples and calculating the average for each service location would reduce the effect of outlier cases and also better meet the anticipated goal of Fresh Express. The final 5 test scenarios were created by generating five 50 samples revenues for each location from the bottom 10% of each distribution and then calculating the average of each 50 sample group. These tests were generated as another goal of Fresh Express is to generate reliable routes. Hence, the routes generated based on these samples are designed to protect against worst case performance in the scenario where all of the service locations experience poor short-term demand.

Using these test scenarios, 12 different customer scenarios were tested and routes were built for each. Specifically, 4 customer scenarios were conducted under each of the following assumptions: Fresh Express cannot satisfy demand at a distance, Fresh Express can satisfy demand up to 400 meters away from their stopping location, and Fresh Express can satisfy demand up to 800 meters away from their stopping location. These different distances (0 meters, 400 meters, and 800 meters) are hereafter referred to as service radii. For each of the radii, different subsets of the service locations were considered:

- All service locations (103 service locations),
- All service locations except schools where less than 50% of students receive free or reduced meals (96 service locations),
- All service location except those with a full service supermarket within 800 meters (75 service locations),
- All service locations except schools where less than 50% of students receive free or reduced meals and those with a full service supermarket within 800 meters (67 service locations).

In summary, 12 different customer scenarios were tested which assume a service radius and a specific subset of customers. Within each customer scenario, 31 different routing plans were generated using the Tabu Search procedure based on the different test scenarios as previously described.

To assist Fresh Express in deciding between these 31 different routing plans per scenario, a simulation approach was conducted to obtain a better understanding of the potential for each of the routing solutions. Specifically, 10,000 samples were taken for each of the 103 service locations and the profit/sample was calculated for each of the 31 routing plans for each of the 12 customer scenarios. Over these samples, the maximum potential profit, minimum potential profit, and average profit were calculated as well as the percentage of samples where the routing plan profit exceeded an established threshold. These values will be cited in the results.

The remaining details refer to settings for the Tabu Search procedure. First, it was assumed that two weeks of routes were to be created as Fresh Express does not frequently visit the same location in consecutive weeks. In addition, it was assumed that each week would have three service days similar to their current operating plan in the Discovery Triangle community. Hence, $|J| = 6$ for these tests. It was assumed that Fresh Express can start operating at 7 AM and must be back to the depot by 6 PM. Also, every school has two time windows defining when service can start, 7:30 AM to 8:30 AM and 2 PM to 3 PM, while every other location can have serviced started from 7 AM to 11 AM and 1 PM to 4 PM. For each location, Fresh Express must stop for 1 hour to fully service the demand. These time requirements match the current operating conditions of Fresh Express.

Since Fresh Express rarely encounters capacity issues, constraint set (7-23) was redefined to represent the limit on the number of stops Fresh Express typically makes during one day. Specifically, Fresh Express only makes four physical stops per day so $C_j = 4$ and $u_m = 1$ for all j and m. Furthermore, since serving a stop at a distance is not a separate stop, any stop whose demand was serviced at a distance does not factor into constraint set (7-23) within the Tabu Search heuristic. However, it was assumed that Fresh Express would only be able to capture 25% of the revenue from covered locations which serves as an estimate for the loss in customers who would not be willing to walk/travel to the vehicle. The travel time between each location was calculated using Manhattan distances assuming an average travel time of 35 mph and then rounded up to the nearest 30 minute interval. This was completed to adhere to the current system used by Fresh Express and to also build in setup and teardown time for Fresh Express at each stop. The cost of travel was based on the distance between all pairs of location and the average fuel cost per mile for CFG buses (Lowell, Chernicoff, and Lian 2007) as the city of Phoenix provides all other minor maintenance costs for free.

The final details are the specific parameters and initialization steps used in the Tabu Search heuristic. For these tests, it was assumed that that $\lambda = 0.015$ and $\eta = 10000$ based on preliminary tests of the algorithm. The lack of extensive testing of these parameters is a clear flaw of this research which will be discussed in Chapter 8. Similar to Cordeau, Laporte, and Mercier (2001), θ was set to $\lfloor 0.5 + 7.5 * \log(|M|) \rfloor$ so that it scales with problem size. To generate the initial routes a simple sweep heuristic was employed similar to the technique used in Chapter 7. Specifically, the service locations were sorted based on their radial angle with the depot and the one service location was selected as the starting point at random. Points were sequentially added to the routes starting at this random service location until every route visited four service locations. Finally, the Tabu Search heuristic was restarted 5 times with $\eta = 10000$ for each test and the routing plan with the highest profit was retained as the final solution for a given test and customer scenario.

A summary of the results for each covering radii are provided in Table 24 through Table 26. For each row, the routing plan which best satisfied the indicated decision criteria (based on the results from the 10000 sample simulation) is provided for each customer scenario. The four decision criteria used in this study are the greatest maximum, greatest minimum, greatest average, greatest percentage of profit observations above \$2100. The limit \$2100 was determined as it is the average two week profit for Fresh Express from the data period. Note this average *excludes* any fuel costs which are included in the profits reported by the model. The largest observation for each statistic within each subset of customers is highlighted in bold.

Table 24. Summary of routing plan statistics assuming 0 meter covering distance for Fresh Express

Cust. Exclusion	Decision Criteria	Maximum Profit	Minimum Profit	Average Profit	% Trials Over $2100
None	Max.	**$4357**	$1154	$2355	73.7%
	Min.	$4010	**$1264**	**$2450**	**81.5%**
	Ave.	$4010	**$1264**	**$2450**	**81.5%**
	Percent.	$4010	**$1264**	**$2450**	**81.5%**
Schools	Max.	**$4257**	$1182	$2383	76.5%
	Min.	$4074	**$1266**	$2343	72.7%
	Ave.	$4127	$1249	**$2418**	**78.5%**
	Percent.	$4127	$1249	**$2418**	**78.5%**
Grocery Vicinity	Max.	**$4439**	$1206	$2313	70.2%
	Min.	$4331	**$1276**	**$2323**	**70.8%**
	Ave.	$4331	**$1276**	**$2323**	**70.8%**
	Percent.	$4331	**$1276**	**$2323**	**70.8%**
Both	Max.	**$3969**	$1130	$2247	64.3%
	Min.	$3821	**$1255**	$2258	64.9%
	Ave.	$3950	$1215	**$2289**	**68.3%**
	Percent.	$3950	$1215	**$2289**	**68.3%**

Table 25. Summary of routing plan statistics assuming 400 meter covering distance for Fresh Express

Cust. Exclusion	Decision Criteria	Maximum Profit	Minimum Profit	Average Profit	% Trials Over $2100
None	Max.	**$4503**	$1283	$2440	80.8%
	Min.	$4448	**$1294**	**$2489**	**83.9%**
	Ave.	$4448	**$1294**	**$2489**	**83.9%**
	Percent.	$4448	**$1294**	**$2489**	**83.9%**
Schools	Max.	**$4411**	$1208	$2407	78.7%
	Min.	$4204	**$1325**	**$2488**	**83.9%**
	Ave.	$4204	**$1325**	**$2488**	**83.9%**
	Percent.	$4204	**$1325**	**$2488**	**83.9%**
Grocery Vicinity	Max.	**$4134**	**$1339**	**$2377**	**76.0%**
	Min.	**$4134**	**$1339**	**$2377**	**76.0%**
	Ave.	**$4134**	**$1339**	**$2377**	**76.0%**
	Percent.	**$4134**	**$1339**	**$2377**	**76.0%**
Both	Max.	**$4069**	$1222	$2303	70.2%
	Min.	$3878	**$1307**	$2299	69.3%
	Ave.	$3918	$1247	**$2320**	71.2%
	Percent.	$3979	$1236	$2318	**71.5%**

Table 26. Summary of routing plan statistics assuming 800 meter covering distance for Fresh Express

Cust. Exclusion	Decision Criteria	Maximum Profit	Minimum Profit	Average Profit	% Trials Over $2100
None	Max.	**$4477**	**$1429**	$2506	86.2%
	Min.	**$4477**	**$1429**	$2506	86.2%
	Ave.	$4428	$1296	**$2562**	**89.0%**
	Percent.	$4428	$1296	**$2562**	**89.0%**
Schools	Max.	**$4581**	$1233	$2481	84.1%
	Min.	$4281	**$1442**	**$2545**	**88.6%**
	Ave.	$4281	**$1442**	**$2545**	**88.6%**
	Percent.	$4281	**$1442**	**$2545**	**88.6%**
Grocery Vicinity	Max.	**$4354**	$1308	$2404	78.9%
	Min.	$4142	**$1355**	$2412	79.7%
	Ave.	$4016	$1281	**$2416**	**79.8%**
	Percent.	$4016	$1281	**$2416**	**79.8%**
Both	Max.	**$4179**	$1213	$2374	76.2%
	Min.	$4134	**$1337**	$2344	74.1%
	Ave.	$4142	$1306	**$2376**	**76.6%**
	Percent.	$4142	$1306	**$2376**	**76.6%**

Shown in Figure 18 is the routing plan which provided the highest average profit for the scenario with a 400 meter covering radius and the subset of service locations which exclude schools where less than 50% of students receive free or reduced meals and those with a full service supermarket within 800 meters. These routes are divided into two panels to reduce the number of overlapping lines. Any service location which was not selected for service in these panels are grey while those which are serviced are black. Any service location serviced at a distance has no lines going to the location, but it is in a filled circle with a visited location which is less than 400 meters away. Also provided is Table 27 which shows the routing plan which had the greatest average profit for each different service radii. The routes in Table 27 assume that the set of potential service locations exclude schools where less than 50% of students receive free or reduced meals and exclude locations with a full service supermarket within 800 meters. Locations which are serviced in all three plans are in bold and locations which are serviced for only one plan are in italics. The first four stops listed for each route are those which are directly visited by Fresh Express while any remaining customers are those which are covered at a distance.

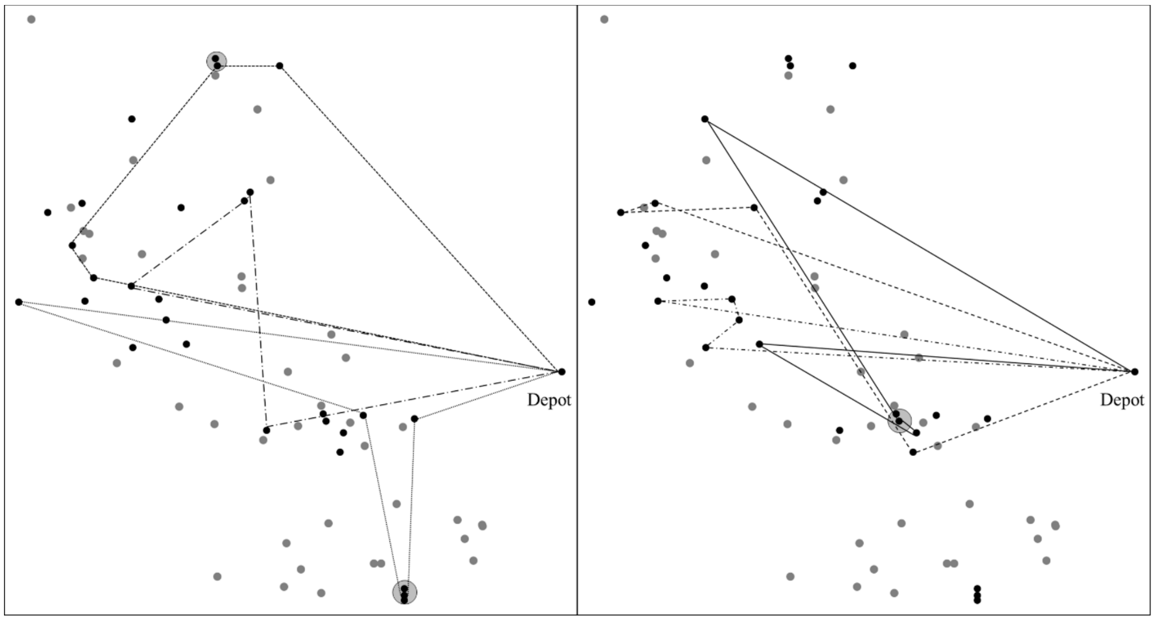


Figure 18. Fresh Express routing plan assuming a 400 meter covering radius and excluding service locations based on school income and supermarket access

Table 27. Fresh Express routing plan excluding service locations based on school income and supermarket access for all covering radius options

	0 M		**400 M**		**800 M**	
Route 1	Orangewood Sc.	7:30	Orangewood Sc.	7:30	Sevilla West Sc.	7:30
	Helen Drake	9:00	**Helen Drake**	9:00	**Helen Drake**	9:00
	Sevilla West Sc.	2:00	Glenn Downs Elem	2:00	Catalina Ventura	2:00
	St. Vincent DePaul	3:30	Urban League Man.	3:30	**Coffelt Housing**	3:30
			Kids Country Club	9:00	Kids Country Club	9:00
					Lemon Grove Apart	9:00
Route 2	*Alhambra Trad.*	7:30	Catalina Ventura	7:30	**Charles W Harris**	7:30
	Kiddie's Kingdom	9:00	**Phoenix Snr Opt.**	9:00	**Maryvale Comm**	9:00
	Alta E Butler Sch	2:00	**Marcos de Niza**	10:30	*Justine Spitalny Sch*	2:00
	Coffelt Housing	3:30	**JB Sutton Elem**	2:00	**Phoenix Snr Opt.**	3:30
			Henson Village	9:00	Glenn Downs Elem	2:00
					Henson Village	3:30
Route 3	Joseph Zito Elem	7:30	Joseph Zito Elem	7:30	**Alta E Butler Sch**	7:30
	B&G Clubs-Colan	9:00	**Kiddie's Kingdom**	9:00	**B&G Clubs-Colan**	9:00
	JB Sutton Elem	2:00	**B&G Clubs-Colan**	10:30	**PT Coe Elem**	10:30
	Victory Place	3:30	**Alta E Butler Sch**	2:00	**Phoenix Day**	2:00
Route 4	**Westwood Prim**	7:30	**PT Coe Elem**	7:30	**Granada Primary**	7:30
	MarcAtkinson Rec	9:00	**MarcAtkinson Rec**	9:00	**MarcAtkinson Rec**	9:00
	Granada Primary	2:00	**Westwood Prim**	2:00	**JB Sutton Elem**	2:00
	Phoenix Snr Opt.	3:30	**Coffelt Housing**	3:30	**St. Vincent DePaul**	3:30
Route 5	**James W Rice**	7:30	**James W Rice**	7:30	**Westwood Prim**	7:30
	Maryvale Comm	9:00	**Maryvale Comm**	9:00	Urban League Man.	9:00
	Charles W Harris	2:00	**Granada Primary**	2:00	**James W Rice**	2:00
	Maricopa Skills	3:30	**St. Vincent DePaul**	3:30	**Kiddie's Kingdom**	3:30
					Andalucia Middle	2:00
Route 6	**CO Greenfield**	7:30	**Charles W Harris**	7:30	*Sabis International*	7:30
	Phoenix Day	9:00	**Phoenix Day**	9:00	Victory Place	9:00
	Marcos de Niza	10:30	**CO Greenfield**	2:00	**CO Greenfield**	2:00
	PT Coe Elem	2:00	Maricopa Skills	3:30	**Marcos de Niza**	3:30
			Amy Houston	2:00	Amy Houston	2:00
			JFK Elementary	2:00	JFK Elementary	2:00

Based on Table 24 through Table 26, it is evident that this solution methodology can develop a wide variety of recommended routing plans. Specifically, the rows in these tables comprise 29 different routing plan within the south and west Phoenix area. Within each table, the performance of the retailer decreases due to the fact that the quantity of excluded customers increase. This result is expected but it demonstrates that the solutions identified by the Tabu Search procedure are likely near the optimal routing plan. The other primary observation is that the factor which has the biggest impact on the profit statistics within each table is whether or not customers who are too close to a supermarket should be excluded. For instance, the difference between the routing plan with all customer available and those which exclude all customers within 800 meters of a supermarket results in $100 average loss across all tables. Hence, Fresh Express should determine their revenue requirements prior to eliminating the service of such customers as they may not be able to meet their goals without servicing these customers.

Table 24 through Table 26 also provide specific observations for Fresh Express. First and foremost, the average profit for each developed routing plan exceeds the current two-week average profit of $2100 reported by Fresh Express. This average even excludes fuel costs, as Fresh Express does not track these costs, thereby implying the developed routes are even better than the current routing plan of Fresh Express. Therefore, by making different routing decisions, it is possible for Fresh Express to increase its economic sustainability.

To recommend a plan from Table 24 through Table 26, Fresh Express must first determine the subset of customers they would like to serve. Clearly the more restrictions Fresh Express wishes to place on its customer base, the less profitable their routes will be but they will be able to better address those most in need. Based on this study, it is highly recommended to only serve those customers which are most in need (i.e. both exclusions are enforced). This is recommended as service locations which are within 800 meters of a supermarket likely feature customers who would prefer to use the supermarket than Fresh Express and service locations which are schools where a majority of students are not low-income are not the target demographic of Fresh Express. For the routes generated within this customer subset, the 400 meter covering radius is recommended for consideration as it is important to consider nearby walking-distance locations when designing Fresh Express service locations, but it is unlikely that customer 800 meters away would be loyal customers. Hence, a 400 meter service radius is a satisfactory balance. Finally, any of the plans within this service radius and customer subset are competitive but selecting the plan which provides the greatest average profit (i.e. the routing plan shown in Figure 18) is recommended as this will provide the greatest longitudinal impact.

The purpose of Figure 18 is three-fold. First, it provides a graphical representation of the recommended routing plan for Fresh Express. Secondly, it provides an example of the routes developed by the Tabu Search procedure. Upon first inspection, the set of six routes in Figure 18 do not appear to represent optimal routes due to the numerous crossings of arcs. However, such crossings are necessary for this case study due to the strict time windows which must be met. Finally, Figure 18 serves to provide an example of the service area and the distribution of possible Fresh Express service locations within the service area. The noticeable gaps in Figure 18 are due to the subset of customers which are excluded as they are too close to supermarkets or due to the Salt River which separates the southerly most 17 service locations from the rest of the network.

Finally, Table 27 shows a subset of routing plans based on the actual serviced (or covered) locations and includes the recommended routing plan. The goal of this table is to demonstrate that the selected covering radius does not have a large effect on the subset of serviced customers. For instance, when a 0 meter service radius is assumed (i.e. the retailer must directly visit a location to service its customers), 18 of the 24 total visited locations are still served in the routing plan from the 400 meter service radius and 800 meter service radius while only 1 of the visited locations is exclusive to the 0 meter service radius. This observation demonstrates that even if the wrong assumption is made regarding the covering radius, it will not drastically effect the solution. Hence, the 400 meter covering radius routing plan was chosen for Fresh Express as it does not drastically differ from the 0 meter covering radius routing plan but it does include the consideration that Fresh Express could pull demand from nearby locations through targeted flyers and advertising.

CHAPTER 8

CONCLUSIONS AND FURTHER RESEARCH

The goal of this dissertation was to address the economic sustainability of mobile healthy food retailers. Such retailers are becoming a popular methodology to alleviate food desert conditions within urban environments, but many are struggling to earn enough revenue to covering their operational and overhead costs. To study these issues, the operational decision making of mobile retailers was addressed by focusing on two problems: the mobile food retailer product mix problem and the mobile food retailer scheduling and routing problem. Models were formulated for these decisions and solution methodologies were developed with specific emphasis on techniques usable by any practitioner. A subset of these tools were then demonstrated using a case study based on a local Phoenix, AZ mobile retailer called Fresh Express.

The conclusions in this section with regards to Chapters 3 through 6 are limited as the end of each chapter featured a summary of future research goals regarding the developed models. Hence, a summary of each of these discussions will be provided in this chapter. More importantly, the findings from the case study will be summarized and future goals with respect to Fresh Express will be discussed. Also discussed in detail are future recommendations for the Tabu Search heuristic which was introduced in Chapter 7.

The purpose of Chapter 3 was to introduce solution methodologies for the MDMKP as this is one modeling approach for the mobile food retailer product mix decision. Specifically, the goal of this chapter was to apply the most advanced KP solver concepts, such as efficiency measures and core variables, to KP variants which had demand constraints. Hence, new efficiency measures were developed which are applicable to any

KP variant with a single linear objective function and linear constraints of any type and quantity. After it was demonstrated that these measures provide the same applicability as traditional efficiency measures, three heuristic solution procedures were presented: a Fixed-Core heuristic, a Kernel Search heuristic, and a Genetic Algorithm heuristic. Each of these techniques have never before applied to the MDMKP and the latter two featured new methodologies for the application of efficiency measures. Future research is recommended to improve each heuristic, especially the Genetic Algorithm with respect to a repair operator, and future research is also recommended to identify new heuristics which may provide superior solutions. With respect to mobile retailers, only the Genetic Algorithm is accessible by all practitioners as it does not require commercial software. However, mobile retailers are recommended to partner with local practitioners who could provide the necessary technical skills to use the higher quality solution methods.

The purpose of Chapter 4 was to introduce a new exact solution methodology for the DKP which is applicable to mobile food retailers who are able to formulate their product mix decision as a two-constraint optimization model. In this chapter, the solution algorithm DKPSOLVE was presented which combines two subroutines for solving the DKP. First, the DKP instance is reduced where variables whose values can be fixed to one of two binary values are removed from the problem. Given these smaller instances, an expanding core procedure is performed which is a depth-first, branch-and-bound process used to determine the final optimal solution. Numerous algorithmic improvements were then detailed for DKPSOLVE prior to extensive computational tests. While DKPSOLVE was already able to outperform commercial software except for the most extreme test instances, numerous research improvements are possible such as improving the data storage techniques and improving the solution heuristics performed inside the reduction procedure. With respect to mobile food retailers, DKPSOLVE is highly applicable as it does not require commercial software, can be provided as a black box program, and has limited data requirements assuming the retailer can formulate their product mix decision using a simplified model.

The purpose of Chapter 5 was to introduce the CCVRP which is used in this dissertation to model the mobile food retailer scheduling and routing problem. Within Chapter 5, the CCVRP formulation is initially provided and an exact solution algorithm is provided which relies on column generation. For this approach the model is transformed into an equivalent set covering formulation and a branch-and-price methodology is used to determine the optimal solution. Columns are generated as needed in this algorithm using a branch-and-bound tree. Extensive computational tests were performed using this algorithm on adapted benchmark cases and problems with up to 50 customers were solved within one hour. Future research is recommended to improve the current column generation procedure by using more advanced subprocedures, such as better TSP solvers, and to develop new approaches such as branch-and-cut algorithms which have shown success for similar problems. With respect to mobile food retailers, the algorithm from this chapter is not directly applicable as it is only useful for small to medium sized problem instances. Instead, the main benefit of this chapter is from serving as a benchmark for the heuristics developed in Chapter 6.

The purpose of Chapter 6 was to outline four heuristics for solving the CCVRP: the Greedy heuristic, the Sweep heuristic, the Savings heuristic, and the ACS heuristic. In this chapter, a route improvement/construction procedure, COVROUTE, was first introduced and then each of the four heuristics were outlined in detail. Thorough tuning tests were then outlined for the ACS heuristic prior to extensive computational tests based on adapted benchmark cases. These results were compared amongst the developed heuristics in Chapter 6 as well as against the exact solution procedure from Chapter 5 when appropriate. With respect to mobile food retailers, each of the developed algorithms are usable by mobile retailers as none require commercial solvers. However, the ACS or Savings heuristic are recommended based on the quality of the tests results.

One of the major contributions from the prior 4 chapters which has yet to be addressed is that these models are applicable to many other types of problems outside of mobile healthy food retailers. For instance, many of the developed techniques are applicable to other mobile retailers such as street trucks and vendors who operate under conditions similar to mobile healthy food retailers. With respect to the developed models, the MDMKP is applicable to a wide array of applications including the project selection problem (Beaujon, Marin, and McDonald 2001), the obnoxious facility location problem (P. Cappanera, Gallo, and Maffioli 2004), and the sea cargo mix problem (Ang, Cao, and Ye 2007). The DKP also has wide ranging applications but the greatest strength of DKPSOLVE is serving as an efficient subprocedure within solution algorithms for more complicated optimization formulations. Finally, the CCVRP also has numerous applications including the routing of mobile signal towers, disaster emergency vehicles, public transportation vehicles, urban school buses, and distribution vehicles servicing smaller secondary distribution hubs from a major centralized facility. The CCVRP can also be used to model evacuation planning in which evacuees can be moved short distances to feasible points where emergency vehicles, such as helicopters, can land and transport patients. The CCVRP can also be used to route objects which are not vehicles. For instance, the CCVRP can be used it to determine the optimal pathing for online visual inspections systems which use cameras to capture details on manufactured circuit boards.

One aspect of the models developed in Chapters 3 through 6 which has not been addressed is that solving the mobile retailer product mix problem and the mobile retailer routing and scheduling problem separately can lead to suboptimal solutions. For instance, it is likely that the demand for items is stop dependent based on the types of customers which live or work around each location. In this research, these problems are treated separately as solving them together is out of scope. However, future research is recommended to investigate whether the models can be combined and solved simultaneously.

Furthermore, it is possible that a hierarchical solution framework can be utilized to develop a heuristic approach to simultaneously optimize both problems. For instance, the product mix problem could be solved given an assumed customer subset. With this solution, the customer demands at each location could be determined for this subset and the routing problem can be solved. With this new routing solution, the product mix can again be addressed given this new customer subset. This problem can be repeated until the solutions converge or a suitable time limit is reached. While this will not optimally solve the problem, the solution will likely be suitable for most mobile retailers.

The purpose of Chapter 7 was to provide a detailed case study, using the Phoenix, AZ retailer Fresh Express, to demonstrate how the operational tools can be employed by mobile retailers as well as to determine whether or not it is possible for such retailers to increase their economic sustainability. Two approaches were utilized to develop a product mix for the mobile retailer. A DKP formulation was used first which was limited only by retailer space and a minimum profit margin while a MDMKP formulation was employed second which factored in constraints to increase the variety of the product mix. Within each approach, the goal of the formulation was to observe the trade-offs between health and consumer costs based on various profit requirements. Data for these tests came from operational data provided by Fresh Express was well as data on additional grocery items from Bashas supermarkets. The results highlighted that numerous product mixes were possible based on the needs of Fresh Express and even conservative mixes could be identified to increase Fresh Express' profits by $100/day with no impact on consumer cost or product healthiness.

The second focus of Chapter 7 was to determine and optimal routing and scheduling plan for Fresh Express through underserved west and south Phoenix locations. To best address the needs of Fresh Express, a new mathematical model was introduced which featured time windows for customer service. A Tabu Search heuristic was then outlined to solve this model and extensive computational experiments were performed to develop Fresh Express routing plans given various restrictions on the customers which could be served and the distance it was assumed customers would travel for service. Data for his study came from Fresh Express routing data for its current customers and all new locations were verified electronically or by in-person interviews. The findings demonstrated that a wide array of routing plans are possible and they all result in average profits which exceed Fresh Express' current earnings. Ultimately, a routing plan was recommended for Fresh Express which served only the most at-risk customers assuming that customers would be willing to travel 0.25 miles at most for service.

In regards to this case study, significant work is recommended with respect to three topics: implementing the product mix recommendations, implementing the routing recommendations, and improving the Tabu Search heuristic. With respect to the Tabu Search heuristic, additional research is recommended on two major issues. Primarily, extensive computational tests have yet to be performed on the Tabu Search heuristic so the ability to measure its performance is not well documented outside of this case study. Secondly, tuning experiments should be performed to determine the ideal settings for the heuristics parameters.

With respect to the product mix, the results from Chapter 7 demonstrated that it is possible to stock a better product mix which earns greater profits for Fresh Express. However, no product mix is necessarily recommended as part of this dissertation as the recommended product mix is at the discretion of Fresh Express and can be determined based on external factors such as required profit margins or similar criteria. Meetings are currently planned with the Executive Director of Fresh Express once the retailer has resumed operations during October as they are currently undergoing maintenance work on the bus. Once the results are disseminated, new supplier options may have to be developed to stock some of the potential items. While waiting for such relationships to be developed, it is expected that new items will be ordered from Peddler’s Son and stocked on Fresh Express to validate the data used in this case study.

With respect to the routing plan developed in Chapter 7, the results will again be disseminated with Fresh Express during a meeting planned later in the year. Based on the findings, it will be recommended that some of the recommended routes be trialed with the current bus for either Monday or Friday service (i.e. a fourth day). To complete this work, new partnerships will have to be developed based on the new service locations. However, as part of this case study, many of the south and west Phoenix were personally contacted and many have already expressed great interest in hosting Fresh Express at their site. Given the ability to trial new routes, data will again be collected from these trials to verify the results from the case study. Based on these results, the data used in the case study can be updated and the routes can be adjusted based on the findings. Once a second bus is added to their system or once new service days are permanently added to their schedule, the developed routes from Chapter 7 can be permanently implemented.

REFERENCES

Agarwal, Y., K. Mathur, and H. Salkin. 1989. “A Set-Partitioning-Based Exact Algorithm for the Vehicle Routing Problem.” *Networks* 19 (7): 731–49.

Akinc, Umit, and Kizhanatham Srikanth. 1992. “Optimal Routing and Process Scheduling for a Mobile Service Facility.” *Networks* 22 (2): 163–83.

Algert, Susan, Aditya Agrawal, and Douglas Lewis. 2006. “Disparities in Access to Fresh Produce in Low-Income Neighborhoods in Los Angeles.” *American Journal of Preventive Medicine* 30 (5): 365–70.

Alkon, Alison Hope, Daniel Block, Kelly Moore, Catherine Gillis, Nicole DiNuccio, and Noel Chavez. 2013. “Foodways of the Urban Poor.” *Geoforum* 48 (1): 126–35.

Alwitt, Linda, and Thomas Donley. 1997. “Retail Stores in Poor Urban Neighborhoods.” *The Journal of Consumer Affairs* 31 (1): 139–64.

Andreyeva, Tatiana, Daniel Blumenthal, Marlene Schwartz, Michael Long, and Kelly Brownell. 2008. “Availability and Prices of Foods Across Stores and Neighborhoods: The Case of New Haven, Connecticut.” *Health Affairs* 27 (5): 1381–88.

Anekwe, Tobenna, and Ilya Rahkovsky. 2013. “Economic Costs and Benefits of Healthy Eating.” *Current Obesity Reports* 2 (3): 225–34.

Ang, James, Chengxuan Cao, and Heng-Qing Ye. 2007. “Model and Algorithms for Multi-Period Sea Cargo Problem.” *European Journal of Operational Research* 180 (3): 1381–93.

Angelelli, Enrico, Renata Mansini, and M. Grazia Speranza. 2010. “Kernel Search: A General Heuristic for the Multi-Dimensional Knapsack Problem.” *Computers & Operations Research* 37 (11). Elsevier: 2017–26.

Arntzen, Halvard, Lars M. Hvattum, and Arne Løkketangen. 2006. “Adaptive Memory Search for Multidemand Multidimensional Knapsack Problems.” *Computer* 33 (9): 2508–25.

Bader, Michael, Marnie Purciel, Paulette Yousefzadeh, and Kathryn Neckerman. 2010. “Disparities in Neighborhood Food Environments: Implications of Measurement Strategies.” *Economic Geography* 86 (4): 409–30.

Baker, Elizabeth, Mario Schootman, Ellen Barnidge, and Cheryl Kelly. 2006. “The Role of Race and Poverty in Access to Foods That Enable Individuals to Adhere to Dietary Guidelines.” *Preventing Chronic Disease* 3 (3): A76.

Balachandar, S., and K. Kannan. 2011. “A New Heuristic Approach for

Knapsack/covering Problem." *International Journal of Mathematical Sciences and Applications* 1 (2): 593–606.

Balas, Egon, and Eitan Zemel. 1980. "An Algorithm for Large Zero-One Knapsack Problems." *Operations Research* 28 (5): 1130–54.

Balsam, Alan, David Webber, and Bonita Oehlke. 1994. "The Farmers' Market Coupon Program for Low-Income Elders." *Journal of Nutrition for the Elderly* 13 (4): 35–42.

Bartholdi III, John, and Steven Hackman. 2011. "Layout of a Piece-Pick-from-Carton Area." In *Warehouse & Distribution Science*, 97–136. Atlanta, GA: The Supply Chain and Logistics Institute.

Beaujon, George, Samuel Marin, and Gary McDonald. 2001. "Balancing and Optimizing a Portfolio of R & D Projects." *Naval Research Logistics* 48 (1): 18–40.

Beaulac, Julie, Elizabeth Kristjansson, and Steven Cummins. 2009. "A Systematic Review of Food Deserts , 1966-2007." *Preventing Chronic Disease* 6 (3): 1–10.

Bell, J, and P McMullen. 2004. "Ant Colony Optimization Techniques for the Vehicle Routing Problem." *Advanced Engineering Informatics* 18 (1): 41–48.

Bertmann, Farryl M W, Punam Ohri-Vachaspati, Matthew P Buman, and Christopher M Wharton. 2012. "Implementation of Wireless Terminals at Farmers' Markets: Impact on SNAP Redemption and Overall Sales." *American Journal of Public Health* 102 (7): e53–55.

Bin, Y, Y Zhong-Zhen, and Y Baozhen. 2009. "An Improved Ant Colony Optimization for Vehicle Routing Problem." *European Journal of Operational Research* 196: 171–76.

Bitler, Marianne, and Steven Haider. 2011. "An Economic View of Food Deserts in the United States." *Policy Retrospectives* 30 (1): 153–76.

Bixby, A., C. Coullard, and D. Simchi-Levi. 1997. "The Capacitated Prize-Collecting Traveling Salesman Problem." Evanston, IL.

Block, Daniel, and Joanne Kouba. 2007. "A Comparison of the Availability and Affordability of a Market Basket in Two Communities in the Chicago Area." *Public Health Nutrition* 9 (7): 837–45.

Blum, Manuel, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan. 1973. "Time Bounds for Selection." *Journal of Computer and Systems Sciences* 7 (4): 448–61.

Bodin, Lawrence D., and Lon Berman. 1979. "Routing and Scheduling of School Buses by Computer.” *Transportation Science* 13 (2): 113–29.

Bodor, J Nicholas, Janet C Rice, Thomas A Farley, Chris M Swalm, and Donald Rose. 2010. “The Association Between Obesity and Urban Food Environments.” *Journal of Urban Health* 87 (5): 771–81.

Bodor, J Nicholas, Donald Rose, Thomas Farley, Christopher Swalm, and Susanne Scott. 2008. “Neighbourhood Fruit and Vegetable Availability and Consumption: The Role of Small Food Stores in an Urban Environment.” *Public Health Nutrition* 11 (4): 413–20.

Boffey, Brian, and Subhash Narula. 1998. “Models for Multi-Path Covering-Routing Problems.” *Annals of Operations Research* 82 (1): 331–42.

Bowerman, Robert, Brent Hall, and Paul Calamai. 1995. “A Multiobjective Optimization Approach to Urban School Bus Routing: Formulation and Solution Method.” *Transportation Research Part A: Policy and Practice* 29 (2): 107–23.

Bullnheimer, B, R Hartl, and C Strauss. 1999. “An Improved Ant System Algorithm for the Vehicle Routing Problem.” *Annals of Operations Research* 89: 319–28.

Bussieck, Michael R., Peter Kreuzer, and Uwe T. Zimmerman. 1996. “Optimal Lines for Railway Systems.” *European Journal of Operational Research* 96: 54–63.

Bussieck, Michael R., Thomas Lindner, and Marco E. Lübbecke. 2004. “A Fast Algorithm for near Cost Optimal Line Plans.” *Mathematical Methods of Operations Research* 59: 205–20.

Califano, Catherine, Kennen Gross, Lance Loethen, Scott Haag, and Ira Goldstein. 2012. “Searching for Markets: The Geography of Inequitable Access to Healthy & Affordable Food in the United States.” The Reinvestment Fund and Opportunity Finance Network.

Cappanera, P., G. Gallo, and F. Maffioli. 2004. “Discrete Facility Location and Routing of Obnoxious Activities.” *Discrete Applied Mathematics* 133 (1): 3–28.

Cappanera, Paola, and Marco Trubian. 2005. “A Local-Search-Based Heuristic for the Demand-Constrained Multidimensional Knapsack Problem.” *INFORMS Journal on Computing* 17 (1): 82–98.

Caraher, Martin, Paul Dixon, Tim Lang, and Roy Carr-Hill. 1998. “Access of Healthy Foods: Part I. Barriers to Accessing Healthy Foods: Differentials by Gender, Social Class, Income and Mode of Transport.” *Health Education Journal* 57 (3): 191–201.

Carlson, Andrea, Mark Lino, WenYen Juan, Kenneth Hanson, and P. Peter Basiotis. 2007. “Thrifty Food Plan, 2006.” United States Department of Agriculture, Center for Nutrition Policy and Promotion.

Caspi, Caitlin, Glorian Sorensen, S V Subramanian, and Ichiro Kawachi. 2012. “The Local Food Environment and Diet: A Systematic Review.” *Health & Place* 18 (5). Elsevier: 1172–87.

Chapleau, Luc, Jacques Ferland, and Jean-Marc Rousseau. 1985. “Clustering for Routing in Densely Populated Areas.” *European Journal of Operational Research* 20: 48–57.

Chu, P.C., and J.E. Beasley. 1998. “A Genetic Algorithm for the Multidimensional Knapsack Problem.” *Journal of Heuristics* 4 (1): 63–86.

Claessens, M. T., N. M. van Dijk, and P. J. Zwaneveld. 1998. “Cost Optimal Allocation of Rail Passenger Lines.” *European Journal of Operational Research* 110: 474–89.

Clarke, G., and J.W. Wright. 1964. “Scheduling of Vehicles from a Central Depot to a Number of Delivery Points.” *Operations Research* 12: 568–81.

Clifton, Kelly. 2004. “Mobility Strategies and Food Shopping for Low-Income Families: A Case Study.” *Journal of Planning Education and Research* 23 (4): 402–13.

Coleman-Jensen, Alisha, Mark Nord, Margaret Andrews, and Steven Carlson. 2012. “Household Food Security in the United States in 2011.”

Cordeau, J-F, G Laporte, and A Mercier. 2001. “A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows.” *The Journal of the Operational Research Society* 52 (8): 928–36.

Cummins, Steven, Mark Petticrew, Cassie Higgins, Anne Findlay, and Leigh Sparks. 2005. “Large Scale Food Retailing as an Intervention for Diet and Health: Quasi-Experimental Evaluation of a Natural Experiment.” *Journal of Epidemiology and Community Health* 59 (12): 1035–40.

Current, John, Hasan Pirkul, and Erik Rolland. 1994. “Efficient Algorithms for Solving the Shortest Covering Path Problem.” *Transportation Science* 28 (4): 317–27.

Current, John, Charles ReVelle, and Jared Cohon. 1984. “The Shortest Covering Path Problem: An Application of Locational Constraints to Network Design.” *Journal of Regional Science* 24 (2): 161–83.

———. 1985. “The Maximum Covering/shortest Path Problem: A Multiobjective Network Design and Routing Formulation.” *European Journal of Operational Research* 21 (2): 189–99.

———. 1987. “The Median Shortest Path Problem: A Multiobjective Approach to Analyze Cost vs. Accessibility in the Design of Transportation Networks.” *Transportation Science* 21 (3): 188–97.

Current, John, and David Schilling. 1989. “The Covering Salesman Problem.” *Transportation Science* 23 (3): 208–13.

———. 1994. “The Median Tour and Maximal Covering Tour Problems: Formulations and Heuristics.” *European Journal of Operational Research* 73 (1): 114–26.

Danaei, Goodarz, Eric Ding, Dariush Mozaffarian, Ben Taylor, Jürgen Rehm, Christopher Murray, and Majid Ezzati. 2009. “The Preventable Causes of Death in the United States: Comparative Risk Assessment of Dietary, Lifestyle, and Metabolic Risk Factors.” *PLoS Medicine* 6 (4): e1000058.

Dantzig, George. 1957. “Discrete-Variable Extremum Problems.” *Operations Research* 5 (2): 266–88.

Della Croce, F., and A. Grosso. 2012. “Improved Core Problem Based Heuristics for the 0/1 Multi-Dimensional Knapsack Problem.” *Computers & Operations Research* 39 (1). Elsevier: 27–31.

Desrochers, Martin, Jacques Desrosiers, and Marius Solomon. 1992. “A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows.” *Operations Research* 40 (2): 342–54.

Diez Roux, Ana, F. Javier Nieto, Laura Caulfield, Hermann Tyroler, Robert Watson, and Moyses Szklo. 1999. “Neighbourhood Difference in Diet: The Atherosclerosis Risk in Communities (ARIC) Study.” *Journal of Epidemiology and Community Health* 53 (1): 55–63.

Ding, Q, X Hu, L Sun, and Y Wang. 2012. “An Improved Ant Colony Optimization and Its Application to Vehicle Routing Problem with Time Windows.” *Neurocomputing* 98: 101–7.

Discovery Triangle. 2016. “Fresh Express.” http://www.discoverytriangle.org/fresh-express/.

Dobson, Gregory. 1982. “Worst-Case Analysis of Greedy Heuristics for Integer Programming with Nonnegative Data.” *Mathematics of Operations Research* 7 (4): 515–31.

Donati, A, R Montemanni, N Casagrande, A E Rizzoli, and L M Gambardella. 2008. “Time Dependent Vehicle Routing Problem with a Multi Ant Colony System.” *European Journal of Operational Research* 185 (3): 1174–91.

Dudziński, Krzysztof, and Stanisław Walukiewicz. 1987. “Exact Methods for the Knapsack Problem and Its Generalizations.” *European Journal of Operational Research* 28 (1): 3–21.

Duran, Cristina. 2007. “Panaderias, Peluquerias, Y Carnicerias: Re-Mexicanizing the

Urban Landscapes of a Southwest City." The University of New Mexico.

Euchi, Jalel, and Rafaa Mraihi. 2012. "The Urban Bus Routing Problem in the Tunisian Case by the Hybrid Artificial Ant Colony Algorithm." *Swarm and Evolutionary Computation* 2: 15–24.

Fan, Lang, and Christine L. Mumford. 2010. "A Metaheuristic Approach to the Urban Transit Routing Problem." *Journal of Heuristics* 16 (3): 353–72.

Favaretto, D, E Moretti, and P Pellegrini. 2007. "Ant Colony System for a VRP with Multiple Time Windows and Multiple Visits." *Journal of Interdisciplinary Mathematics* 10 (2): 263–84.

Fisher, M.L., and R. Jaikumar. 1981. "A Generalized Assignment Heuristics for Vehicle Routing." *Networks* 11: 109–24.

Flamm, Laura. 2011. "Barriers to EBT Use at Farmers' Markets: Lessons in Empowerment Evaluation from Rural Ohio." *Journal of Hunger & Environmental Nutrition* 6 (1): 54–63.

Flegal, Katherine, Margaret Carroll, Brian Kit, and Cynthia Ogden. 2012. "Prevalence of Obesity and Trends in the Distribution of Body Mass Index Among US Adults, 1999-2010." *Journal of the American Medical Association* 307 (5): 491–97.

Food, Conservation, and Energy Act of 2008. 2008. H.R. 2419.

Gajpal, Y, and P Abad. 2009. "An Ant Colony System (ACS) for Vehicle Routing Problem with Simultaneous Delivery and Pickup." *Computers & Operations Research* 36 (12): 3215–23.

Gendreau, Michel, Gilbert Laporte, and JY Potvin. 2001. "Metaheuristics for the Capacitated VRP." In *The Vehicle Routing Problem*, edited by Paolo Toth and Daniele Vigo, 129–54. Philadelphia: Society for Industrial and Applied Mathematics.

Ghasemi, Taha, and Mohammadreza Razzazi. 2011. "Development of Core to Solve the Multidimensional Multiple-Choice Knapsack Problem." *Computers & Industrial Engineering* 60 (2): 349–60.

Gillett, B, and L Miller. 1974. "A Heuristic Algorithm for the Vehicle Dispatch Problem." *Operations Research* 22: 340–49.

Gomes da Silva, Carlos, João Clímaco, and José Rui Figueira. 2008. "Core Problems in Bi-Criteria {0, 1}-Knapsack Problems." *Computers & Operations Research* 35 (7): 2292–2306.

Goossens, Jan-Willem, Stan van Hoesel, and Leo Kroon. 2004. "A Branch-and-Cut

Approach for Solving Railway Line-Planning Problems.” *Transportation Science* 38 (3): 379–93.

Grace, Christina, Thomas Grace, Nancy Becker, and Judy Lyden. 2007. “Barriers to Using Urban Farmers’ Markets: An Investigation of Food Stamp Clients’ Perceptions.” *Journal of Hunger & Environmental Nutrition* 2 (1): 55–75.

Grashof, John. 1970. “Supermarket Chain Product Mix Decision Criteria: A Simulation Experiment.” *Journal of Marketing Research* 7 (2): 235–42.

Gu, J., M. Goetschalckx, and L.F. McGinnis. 2010. “Solving the Forward-Reserve Allocation Problem in Warehouse Order Picking Systems.” *Journal of the Operational Research Society* 61 (6): 1013–21.

Gustafson, Alison, Sarah Lewis, Kate Moore, and Stephanie Jilcott. 2013. “Food Venue Choice, Consumer Food Environment, but Not Food Venue Availability within Daily Travel Patterns Are Associated with Dietary Intake Among Adults, Lexington Kentucky 2011.” *Nutrition Journal* 12 (1): 17.

Hackman, Steven, and Loren Platzman. 1990. “Near-Optimal Solution of Generalized Resource Allocation Problems with Large Capacities.” *Operations Research* 38 (5): 902–10.

Hackman, Steven, Meir Rosenblatt, and John Olin. 1990. “Allocating Items to an Automated Storage and Retrieval System.” *IIE Transactions* 22 (1): 7–14.

Halper, Russell, and Srinivasa Raghavan. 2011. “The Mobile Facility Routing Problem.” *Transportation Science* 45 (3): 413–34.

Hansen, Pierre, and Hans Heinsbroek. 1979. “Product Selection and Space Allocation in Supermarkets.” *European Journal of Operational Research* 3 (6): 474–84.

Haynes-Maslow, Lindsey, Sarah Parsons, Stephanie Wheeler, and Lucia Leone. 2013. “A Qualitative Study of Perceived Barriers to Fruit and Vegetable Consumption Among Low-Income Population, North Carolina, 2011.” *Preventing Chronic Disease* 10 (E34).

Heeler, Roger, Michael Kearney, and Bruce Mehaffey. 1973. “Modeling Supermarket Product Selection.” *Journal of Marketing Research* 10 (1): 34–37.

Heragu, S.S., L. Du, R.J. Mantel, and P.C. Schuur. 2005. “Mathematical Model for Warehouse Design and Product Allocation.” *International Journal of Production Research* 43 (2): 327–38.

Hvattum, Lars, Halvard Arntzen, Arne Løkketangen, and Fred Glover. 2010. “Alternating Control Tree Search for Knapsack/covering Problems.” *Journal of Heuristics* 16 (3): 239–58.

Hvattum, Lars M., and Arne Løkketangen. 2007. “Experiments Using Scatter Search for the Multidemand Multidimensional Knapsack Problem.” In *Metaheuristics: Progress in Complex System Optimization*, edited by Karl Doerner, Michel Gendreau, Peter Greistorfer, Walter Gutjahr, Richard Hartl, and Marc Reimann, 3–24. New York, NY: Springer Science+Business Media.

Kaufman, Phillip, James MacDonald, Steve Lutz, and David Smallwood. 1997. “Do the Poor Pay More for Food? Item Selection and Price Differences Affect Low-Income Household Food Costs.” *Monographs of the Society for Research in Child Development*. Food and Rural Economics Division, Economic Research Service.

Kong, Chenying, and Dale Masel. 2008. “Methods for Design and Management of a Fast-Pick Area in a Warehouse.” In *Proceedings of the 2008 Industrial Engineering Research Conference*, edited by J. Fowler and Mason S., 1367–72. Vancouver, Canada.

Laporte, Gilbert, Juan A. Mesa, Francisco A. Ortega, and Ignacio Sevillano. 2005. “Maximizing Trip Coverage in the Location of a Single Rapid Transit Alignment.” *Annals of Operations Research* 136: 49–63.

Laporte, Gilbert, and Frédéric Semet. 2014. “Classical Heuristics for the Capacitated VRP.” In *Vehicle Routing: Problems, Methods, and Applications*, edited by Paolo Toth and Daniele Vigo, 2nd ed., 109–28. SIAM.

Larsen, Kristian, and Jason Gilliland. 2008. “Mapping the Evolution of ‘Food Deserts’ in a Canadian City: Supermarket Accessibility in London, Ontario, 1961-2005.” *International Journal of Health Geographics* 7 (16).

Larson, Nicole, Mary Story, and Melissa Nelson. 2009. “Neighborhood Environments: Disparities in Access to Healthy Foods in the U.S.” *American Journal of Preventive Medicine* 36 (1). American Journal of Preventive Medicine: 74–81.

Leone, Lucia A, Diane Beth, Scott B Ickes, Kathleen Macguire, Robert Andrew Smith, Deborah F Tate, and Alice S Ammerman. 2012. “Attitudes Toward Fruit and Vegetable Consumption and Farmers’ Market Usage Among Low-Income North Carolinians.” *Journal of Hunger & Environmental Nutrition* 7 (1): 64–76.

Lopez, Russ. 2007. “Risk Factors and Chronic Disease Neighborhood Risk Factors for Obesity.” *Obesity* 15 (8): 2111–19.

Lowell, Dana, William Chernicoff, and F. Scott Lian. 2007. “Fuel Cell Bus Life Cycle Cost Model: Base Case & Future Scenario Analysis.”

Lust, Thibaut, and Jacques Teghem. 2012. “The Multiobjective Multidimensional Knapsack Problem: A Survey and a New Approach.” *International Transactions in Operational Research* 19 (4): 495–520.

Martello, Silvano, David Pisinger, and Paolo Toth. 1999. “Dynamic Programming and Strong Bounds for the 0-1 Knapsack Problem.” *Management Science* 45 (3): 414–24.

———. 2000. “New Trends in Exact Algorithms for the 0-1 Knapsack Problem.” *European Journal of Operational Research* 123 (2): 325–32.

Martello, Silvano, and Paolo Toth. 1990. “An Exact Algorithm for Large Unbounded Knapsack Problems.” *Operations Research Letters* 9 (1): 15–20.

———. 1997. “Upper Bound and Algorithms for Hard 0-1 Knapsack Problems.” *Operations Research* 45 (5): 768–78.

———. 2003. “An Exact Algorithm for the Two-Constraint 0-1 Knapsack Problem.” *Operations Research* 51 (5): 826–35.

Mavrotas, George, José Rui Figueira, and Kostas Florios. 2009. “Solving the Bi-Objective Multi-Dimensional Knapsack Problem Exploiting the Concept of Core.” *Applied Mathematics and Computation* 215 (7). Elsevier Inc.: 2502–14.

McGuirt, Jared T, Stephanie B Jilcott, Haiyong Liu, and Alice S Ammerman. 2011. “Produce Price Savings for Consumers at Farmers’ Markets Compared to Supermarkets in North Carolina.” *Journal of Hunger & Environmental Nutrition* 6 (1): 86–98.

McKinnon, Robin, Jill Reedy, Meredith Morrissette, Leslie Lytle, and Amy Yaroch. 2009. “Measures of the Food Environment: A Compilation of the Literature, 1990-2007.” *American Journal of Preventive Medicine* 36 (4 Suppl): S124–33.

Megiddo, Nimrod, and Arie Tamir. 1993. “Linear Time Algorithms for Some Separable Quadratic Programming Problems.” *Operations Research Letters* 13 (4): 203–11.

Mohaymany, A. Shariat, and N. Pirnazar. 2007. “Critical Routes Determination for Emergency Transportation Network Aftermath Earthquake.” In *Proceedings of the 2007 IEEE IEEM*, edited by Andrew Nee, Hans-Otto Guenther, Robert Grubbstrom, Kien Ming Ng, Aarnout Brombacher, and Way Kuo, 817–21. Singapore.

Montgomery, Douglas. 2012. *Design and Analysis of Experiments*. 8th ed. Hoboken, New Jersey: Wiley.

Moore, Latetia, and Ana Diez Roux. 2006. “Associations of Neighborhood Characteristics with the Location and Type of Food Stores.” *American Journal of Public Health* 96 (2): 325–31.

Morland, Kimberly, Steve Wing, and Ana Diez Roux. 2002. “The Contextual Effect of the Local Food Environment on Residents’ Diets: The Atherosclerosis Risk in Communities Study.” *American Journal of Public Health* 92 (11): 1761–67.

Morland, Kimberly, Steve Wing, Ana Diez Roux, and Charles Poole. 2002. "Neighborhood Characteristics Associated with the Location of Food Stores and Food Service Places." *American Journal of Preventive Medicine* 22 (1): 23–29.

Nelson, Melissa, Penny Gordon-Larsen, Kari North, and Linda Adair. 2006. "Body Mass Index Gain, Fast Food, and Physical Activity: Effects of Shared Environments over Time." *Obesity* 14 (4): 701–9.

Network and Emerging Optimization Research Group. 2013. "Capacitated VRP Instances." http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/.

Ohls, James, Michael Ponza, Lorenzo Moreno, Amy Zambrowski, and Rhoda Cohen. 1999. "Food Stamp Participants ' Access to Food Retailers." Princeton, NJ: Mathematica Policy Research, Inc.

Park, Junhyuk, and Byung-In Kim. 2010. "The School Bus Routing Problem: A Review." *European Journal of Operational Research* 202: 311–19.

Pattnaik, S. B., S. Mohan, and V. M. Tom. 1998. "Urban Bus Transit Route Network Design Using Genetic Algorithm." *Journal of Transportation Engineering* 124 (4): 368–75.

Pferschy, Ulrich, and Rostislav Staněk. 2014. "Generating Subtour Constraints for the TSP from the Pure Integer Solutions." Graz, Austria.

Pisinger, David. 1995a. "An Expanding-Core Algorithm for the Exact 0-1 Knapsack Problem." *European Journal of Operational Research* 87 (1): 175–87.

———. 1995b. "A Minimal Algorithm for the Multiple-Choice Knapsack Problem." *European Journal of Operational Research* 83 (2): 394–410.

———. 1997. "A Minimal Algorithm for the 0-1 Knapsack Problem." *Operations Research* 45 (5): 758–67.

———. 2000. "A Minimal Algorithm for the Bounded Knapsack Problem." *INFORMS Journal on Computing* 12 (1): 75–82.

Poggi, Marcus, and Eduardo Uchoa. 2014. "New Exact Algorithms for the Capacitated Vehicle Routing Problem." In *Vehicle Routing: Problems, Methods, and Applications*, edited by Paolo Toth and Daniele Vigo, 2nd ed., 59–86. SIAM.

Powell, Lisa, M. Christopher Auld, Frank Chaloupka, Patrick O'Malley, and Lloyd Johnston. 2007. "Associations Between Access to Food Stores and Adolescent Body Mass Index." *American Journal of Preventive Medicine* 33 (Suppl): S301–7.

Powell, Lisa, Sandy Slater, Donka Mirtcheva, Yanjun Bao, and Frank Chaloupka. 2007. "Food Store Availability and Neighborhood Characteristics in the United States."

Preventive Medicine 44 (3): 189–95.

Puchinger, Jakob, Günther R. Raidl, and Ulrich Pferschy. 2010. "The Multidimensional Knapsack Problem: Structure and Algorithms." *INFORMS Journal on Computing* 22 (2): 250–65.

Racine, Elizabeth F, Ashley Smith Vaughn, and Sarah B Laditka. 2010. "Farmers' Market Use Among African-American Women Participating in the Special Supplemental Nutrition Program for Women, Infants, and Children." *Journal of the American Dietetic Association* 110 (3). Elsevier Inc.: 441–46.

Reeves, Mathew, and Ann Rafferty. 2005. "Healthy Lifestyle Characteristics Among Adults in the United States, 2000." *Archives of Internal Medicine* 165 (8): 854–57.

Renaud, J, F Boctor, and G Laporte. 1996. "An Improved Petal Heuristic for the Symmetric Routing Problem." *Journal of the Operational Research Society* 47: 329–36.

Reyes, Pedro, and Gregory Frazier. 2007. "Goal Programming Model for Grocery Shelf Space Allocation." *European Journal of Operational Research* 181 (2): 634–44.

Riera-Ledesma, Jorge, and Juan-José Salazar-González. 2012. "Solving School Bus Routing Using Multiple Vehicle Traveling Purchaser Problem: A Branch-and-Cut Approach." *Computers & Operations Research* 39: 391–404.

Robert, Stephanie, and Eric Reither. 2004. "A Multilevel Analysis of Race, Community Disadvantage, and Body Mass Index Among Adults in the US." *Social Science & Medicine* 59 (12): 2421–34.

Rose, Donald, J. Nicholas Bodor, Chris Swalm, Janet Rice, Thomas Farley, and Paul Hutchinson. 2009. "Deserts in New Orleans? Illustrations of Urban Food Access and Implications for Policy." Ann Arbor, MI: University of Michigan National Poverty Center/USDA Economic Research Service Research.

Rose, Donald, and Rickelle Richards. 2004. "Food Store Access and Household Fruit and Vegetable Use Among Participants in the US Food Stamp Program." *Public Health Nutrition* 7 (8): 1081–88.

Ruelas, Valerie, Ellen Iverson, Preston Kiekel, and Anne Peters. 2012. "The Role of Farmers' Markets in Two Low Income, Urban Communities." *Journal of Community Health* 37 (3): 554–62.

Rundle, Andrew, Ana Diez Roux, Lance Freeman, Douglas Miller, Kathryn Neckerman, and Christopher Weiss. 2007. "The Urban Built Environment and Obesity in New York City: A Multilevel Analysis." *American Journal of Health Promotion* 21 (4): 326–34.

Salkin, Harvey, and Cornelis De Kluyver. 1975. “The Knapsack Problem: A Survey.” *Naval Research Logistics* 22 (1): 127–44.

Schittekat, Patrick, Joris Kinable, Kenneth Sörenson, Marc Sevaux, Frits Spieksma, and Johan Springael. 2013. “A Metaheuristic for the School Bus Routing Problem with Bus Stop Selection.” *European Journal of Operational Research* 229: 518–28.

Schittekat, Patrick, Marc Sevaux, and Kenneth Sörenson. 2006. “A Mathematical Formulation for a School Bus Routing Problem.” In *2006 International Conference on Service Systems and Service Management*, 1552–57. IEEE.

Schöbel, Anita. 2012. “Line Planning in Public Transportation: Models and Methods.” *OR Spectrum* 34: 491–510.

Semet, Frédéric, Paolo Toth, and Daniele Vigo. 2014. “Classical Exact Algorithms for the Capacitated Vehicle Routing Problem.” In *Vehicle Routing: Problems, Methods, and Applications*, edited by Paolo Toth and Daniele Vigo, 2nd ed., 37–57. SIAM.

Soumis, Francois, Jacques Desrosiers, and Martin Desrochers. 1984. “Optimal Urban Bus Routing with Scheduling Flexibilities.” In *System Modelling and Optimization*, 155–65. Springer Berlin Heidelberg.

Stimpson, Jim, Hyunsu Ju, Mukaila Raji, and Karl Eschbach. 2007. “Neighborhood Deprivation and Health Risk Behaviors in NHANES III.” *American Journal of Health Behavior* 31 (2): 215–22.

Taillard, E. 1993. “Parallel Iterative Search Methods for Vehicle Routing Problems.” *Networks* 23: 661–73.

van den Berg, Jeroen, Gunter Sharp, A.J.R.M. Gademann, and Yves Pochet. 1998. “Forward-Reserve Allocation in a Warehouse with Unit-Load Replenishments.” *European Journal of Operational Research* 111 (1): 98–113.

Ver Ploeg, Michele, Vince Breneman, Tracey Farrigan, Karen Hamrick, David Hopkins, Phillip Kaufman, Biing-Hwan Lin, et al. 2009. “Access to Affordable and Nutritious Food: Measuring and Understanding Food Deserts and Their Consequences Report to Congress.” USDA Economic Research Service.

Ver Ploeg, Michele, and Paula Dutko. 2013. “Food Access Research Atlas Documentation.” *U.S. Department of Agriculture, Economic Research Service*. http://www.ers.usda.gov/data-products/food-access-research-atlas/documentation.aspx#.UyNg-vmwI0q.

Volgenant, A., and S. Marsman. 1998. “A Core Approach to the 0-1 Equality Knapsack Problem.” *Journal of the Operational Research Society* 49 (1): 86–92.

Walker, Renee, Christopher Keane, and Jessica Burke. 2010. “Disparities and Access to

Healthy Food in the United States: A Review of Food Deserts Literature." *Health & Place* 16 (5). Elsevier: 876–84.

Weatherspoon, Dave, James Oehmke, Assa Dembélé, Marcus Coleman, Thasanee Satimanon, and Lorraine Weatherspoon. 2013. "Price and Expenditure Elasticities for Fresh Fruits in an Urban Food Desert." *Urban Studies* 50 (1): 88–106.

Widener, Michael, Sara Metcalf, and Yaneer Bar-Yam. 2012. "Developing a Mobile Produce Distribution System for Low-Income Urban Residents in Food Deserts." *Journal of Urban Health* 89 (5): 733–45.

———. 2013. "Agent-Based Modeling of Policies to Improve Urban Food Access for Low-Income Populations." *Applied Geography* 40 (1). Elsevier Ltd: 1–10.

Wilbaut, Christophe, Said Hanafi, and Said Salhi. 2008. "A Survey of Effective Heuristics and Their Application to a Variety of Knapsack Problems." *IMA Journal of Management Mathematics* 19 (3): 227–44.

Wishon, Christopher, and J. Rene Villalobos. 2016a. "Alleviating Food Disparities with Mobile Retailers: Dissecting the Problem from an OR Perspective." *Computers & Industrial Engineering* 91: 154–64.

———. 2016b. "Robust Efficiency Measures for Linear Knapsack Problem Variants." *European Journal of Operational Research* 254: 398–409.

Wrigley, Neil, Daniel Warm, and Barrie Margetts. 2003. "Deprivation, Diet, and Food-Retail Access: Finding from the Leeds 'Food Deserts' Study." *Environment and Planning A* 35 (1): 151–88.

Wu, Changshan, and Alan Murray. 2005. "Optimizing Public Transit Quality and System Access: The Multiple-Route, Maximal Covering/shortest-Path Problem." *Environment and Planning B: Planning and Design* 32 (2): 163–78.

Yan, Shangyao, and Hao-Lei Chen. 2002. "A Scheduling Model and a Solution Algorithm for Inter-City Bus Carriers." *Transportation Research Part A: Policy and Practice* 36 (9): 805–25.

Zenk, Shannon, Amy Schulz, Srimathi Kannan, Laurie Lachance, Graciela Mentz, and William Ridella. 2009. "Neighborhood Retailer Food Environment and Fruit and Vegetable Intake in a Multiethnic Urban Population." *American Journal of Health Promotion* 23 (4): 255–64.

Zepeda, Lydia. 2009. "Which Little Piggy Goes To Market? Characteristics of US Farmers' Market Shoppers." *International Journal of Consumer Studies* 33 (3): 250–57.

Zhang, X, and L Tang. 2009. "A New Hybrid Ant Colony Optimization Algorithm for the Vehicle Routing Problem." *Pattern Recognition Letters* 30: 848–55.

Zufryden, Fred. 1986. "A Dynamic Programming Approach for Product Selection and Supermarket Shelf-Space Allocation." *The Journal of the Operational Research Society* 37 (4): 413–22.

APPENDIX A

DKPSOLVE PSUEDOCODE

The pseudocode for the IMP procedure follows. As input, $\boldsymbol{x} = \{x_1, \dots, x_n\}$ is a solution for the current DKP which is not necessarily feasible, W and V are the current left hand side values for (4-2) and (4-3) respectively, and f is a Boolean indicator of feasibility. Note that the variables in $\boldsymbol{x}$ are sorted according to their efficiency measures. As output, $\boldsymbol{x}$, W, V, and f are updated and $\boldsymbol{x}$ is at least feasible with respect to (4-2) and possibly is feasible with respect to (4-3). The objective of IMP is to take the current vector $\boldsymbol{x}$ and first ensure feasibility with respect to (4-2) prior to greedily filling the remainder of the solution with emphasis first on obtaining feasibility of (4-3) and second on solution quality.

```
**procedure** IMP($\boldsymbol{x}$, $W$, $V$, $f$)
    $j = n + 1$;
    *if* $f = 0$ and $W > C$ *then*
        *while* $W > C$ *do*
            $k =$ largest index $< j$ such that $x_k = 1$;
            $x_k = 0, W = W - w_k, V = V - v_k, j = k$;
        *end while*
        *if* $V \geq R$ *then* $f = 1$;
    *end if*
    *for* $j = 1$ to $n$ *do*
        *if* $x_j = 0$ *then*
            *if* $f = 0$ and $W + w_j \leq C$ *then*
                $x_j = 1, W = W + w_j, V = V + v_j$;
                *if* $W \leq C$ and $V \geq R$ *then* $f = 1$;
            *else if* $f = 1$, $W + w_j \leq C$, and $p_j > 0$ *then* $x_j = 1$, $W = W + w_j$, $V = V + v_j$;
        *end if*
    *end for*
*end*
```

The pseudocode for the REMREPL procedure follows. As input, $\boldsymbol{x} = \{x_1, \dots, x_n\}$ is a solution for the current DKP which is guaranteed to be feasible with respect to (4-2), W and V are the current left hand side values for (4-2) and (4-3) respectively, P is the value of (4-1), f is a Boolean indicator of feasibility, and a is a global parameter. Note that the variables in $\boldsymbol{x}$ are sorted according to their efficiency measures. As output, $\boldsymbol{x}$, W, V, P, and f are updated and $\boldsymbol{x}$ is at least feasible with respect to (4-2) and possibly is feasible with respect to (4-3). The objective of REMREPL is to sequentially test the separate removal of up to a included items from the solution prior to greedily filling the remainder of the solution with emphasis first on obtaining feasibility of (4-3) and second on solution quality.

```
**procedure** REMREPL($\boldsymbol{x}$, $P$, $W$, $V$, $f$, $a$)
    $P' = P, W' = W, V' = V, k =$ smallest index such that $x_j = 0, f' = f$;
    *for* $j = k + 1$ to $n$ *do if* $x_j = 1$ *then*
        $P' = P' - p_j, W' = W' - w_j, V' = V' - v_j$;
    $n' \coloneqq \min(k - 1, a), z \coloneqq 0, i' = \emptyset$;
    *for* $j = k - 1$ down to $k - n'$ *do*
        $P'' = P' - p_j, W'' = W' - w_j, V'' = V' - v_j$;
        *for* $l = k$ to $n$ *do*
            *if* $W'' + w_l \leq C$ and $V'' < R$ *then*
                $P'' = P' + p_l, W'' = W' + w_l, V'' = V' + v_l$;
            *else if* $W'' + w_l \leq C, V'' \geq R$ and $p_l > 0$ *then*
                $P'' = P' + p_l, W'' = W' + w_l, V'' = V' + v_l$;
        *end for*
        *if* $f = 0$ and $V'' \geq R$ *then* $z = P'' - P, i' = j, f = 1$;
        *else if* $f = 1$ and $V'' \geq R$ and $P'' - P > z$ *then* $z = P'' - P, i' = j$;
    *end for*
    *if* $z \neq 0$ or $f' \neq f$ *then*
        $f = 1, P = P + z, W = W' - w_{i'}, V = V' - v_{i'}$;
        $x_{i'} = 0$;
        *for* $j = k$ to $n$ *do*
            $x_j = 0$;
            *if* $W + w_j \leq C$ and $V < R$ *then*
                $W = W + w_j, V = V + v_j, x_j = 1$;
            *else if* $W + w_j \leq C, V \geq R$ and $p_j > 0$ *then*
                $W = W + w_j, V = V + v_j, x_j = 1$;
        *end for*
    *end if*
*end*
```

The pseudocode for the REMREPL2 procedure follows. As input, $\boldsymbol{x} = \{x_1, \dots, x_n\}$ is a solution for the current DKP which is guaranteed to be feasible with respect to (4-2), W and V are the current left hand side values for (4-2) and (4-3) respectively, P is the value of (4-1), f is a Boolean indicator of feasibility, and a' is a global parameter. Note that the variables in $\boldsymbol{x}$ are sorted according to their efficiency measures. As output, $\boldsymbol{x}$, W, V, P, and f are updated and $\boldsymbol{x}$ is at least feasible with respect to (4-2) and possibly is feasible with respect to (4-3). The objective of REMREPL2 is to sequentially test the separate removal of included pairs of items from the solution prior to greedily filling the remainder of the solution with emphasis first on obtaining feasibility of (4-3) and second on solution quality.

```
**procedure** REMREPL2($\boldsymbol{x}$, $P$, $W$, $V$, $f$, $a'$)
    $P' = P$, $W' = W$, $V' = V$, $k =$ smallest index such that $x_j = 0$, $f' = f$;
    *for* $j = k + 1$ to $n$ *do if* $x_j = 1$ *then*
        $P' = P' - p_j$, $W' = W' - w_j$, $V' = V' - v_j$;
    $n' \coloneqq \min(k - 1, a')$, $z \coloneqq 0$, $i' = \emptyset$, $i'' = \emptyset$;
    *for* $j = k - 1$ down to $k - n' + 1$ *do*
        *for* $j' = j - 1$ down to $k - n'$ *do*
            $P'' = P' - p_j - p_{j'}$, $W'' = W' - w_j - w_{j'}$, $V'' = V' - v_j - v_{j'}$;
            *for* $l = k$ to $n$ *do*
                *if* $W'' + w_l \leq C$ and $V'' < R$ *then*
                    $P'' = P' + p_l$, $W'' = W' + w_l$, $V'' = V' + v_l$;
                *else if* $W'' + w_l \leq C$, $V'' \geq R$ and $p_l > 0$ *then*
                    $P'' = P' + p_l$, $W'' = W' + w_l$, $V'' = V' + v_l$;
            *end for*
            *if* $f = 0$ and $V'' \geq R$ *then* $z = P'' - P$, $i' = j$, $i'' = j'$, $f = 1$;
            *else if* $f = 1$ and $V'' \geq R$ and $P'' - P > z$ *then* $z = P'' - P$, $i' = j$, $i'' = j'$;
        *end for*
    *end for*
    *if* $z \neq 0$ or $f' \neq f$ *then*
        $f = 1$, $P = P + z$, $W = W' - w_{i'} - w_{i''}$, $V = V' - v_{i'} - v_{i''}$;
        $x_{i'} = 0$, $x_{i''} = 0$;
        *for* $j = k$ to $n$ *do*
            $x_j = 0$;
            *if* $W + w_j \leq C$ and $V < R$ *then*
```

```
                $W = W + w_j, V = V + v_j, x_j = 1;$
            *else if* $W + w_j \leq C, V \geq R$ and $p_j > 0$ *then*
                $W = W + w_j, V = V + v_j, x_j = 1;$
        *end for*
    *end if*
*end*
```

The pseudocode for the FEAS procedure follows. As input, $\boldsymbol{x} = \{x_1, \dots, x_n\}$ is a solution for the current DKP which is guaranteed to be feasible with respect to (4-2), W and V are the current left hand side values for (4-2) and (4-3) respectively, P is the value of (4-1), and f is a Boolean indicator of feasibility. Note that the variables in $\boldsymbol{x}$ are sorted according to non-increasing values of v_i/w_i with ties broken according to p_i. As output, $\boldsymbol{x}$, W, V, P, and f are updated and $\boldsymbol{x}$ is guaranteed to be feasible so long as such a solution exists.

```
**procedure** FEAS($\boldsymbol{x}$, $P$, $W$, $V$, $f$)
    $P = 0, W = 0, V = 0;$
    *for* $j = i$ to $n$ *do*
        $x_j = 0;$
        *if* $W + w_j \leq C$ and $V < R$ *then*
            $P = P + p_j \; W = W + w_j, V = V + v_j, x_j = 1;$
        *else* if $W + w_j \leq C, V \geq R$ and $p_j > 0$ *then*
            $P = P + p_j \; W = W + w_j, V = V + v_j, x_j = 1;$
    *end for*
    *if* $V \geq R$ *then* $f = 1;$
*end*
```

The pseudocode for the REDUCE procedure follows. As input, $\boldsymbol{x} = \{x_1, \dots, x_n\}$ is storage for the best solution for the current DKP, C and R are the limits of (4-2) and (4-3) respectively, and n_s is an integer indicating the index of the first variable to be investigated.

```
**procedure** REDUCE($\boldsymbol{x}$, $C$, $R$, $n_s$)
    initialize $\boldsymbol{x}$, $P$, $W$, $V$, and $f$;
    $U^* = \min\left(\left\lfloor z\left(L_R(DKP,\tilde{\lambda})\right)\right\rfloor, \lfloor z(L_C(DKP,\tilde{\mu}))\rfloor, z\left(S(DKP,\tilde{\alpha},\tilde{\beta})\right)\right)$;
    IMP($\boldsymbol{x}$, $W$, $V$, $f$), REMREPL($\boldsymbol{x}$, $P$, $W$, $V$, $f$, $a$), REMREPL2($\boldsymbol{x}$, $P$, $W$, $V$, $f$, $a'$);
    *if* $f = 0$ *then*
        FEAS($\boldsymbol{x}$, $P$, $W$, $V$, $f$);
        *if* $f = 0$ *then* break;
        IMP ($\boldsymbol{x}$, $W$, $V$, $f$), REMREPL ($\boldsymbol{x}$, $P$, $W$, $V$, $f$, $a$), REMREPL2($\boldsymbol{x}$, $P$, $W$, $V$, $f$, $a'$);
    *end if*
    *if* $U^* = P$ *then* break;
    *else*
        $j = n_s$;
        *do*
            *if* $(p_j + \tilde{\lambda}v_j)/w_i \geq \left(p_{b(\tilde{\lambda})} + \tilde{\lambda}v_{b(\tilde{\lambda})}\right)/w_{b(\tilde{\lambda})}$ *then*
                *if* $\lfloor CLU_R(\tilde{\lambda})\rfloor_{x_j=0} \leq P$ *then* $I_1 = I_1 \cup \{i\}$;
            *if* $(p_j + \tilde{\lambda}v_j)/w_i \leq \left(p_{b(\tilde{\lambda})} + \tilde{\lambda}v_{b(\tilde{\lambda})}\right)/w_{b(\tilde{\lambda})}$ *then*
                *if* $\lfloor CLU_R(\tilde{\lambda})\rfloor_{x_j=1} \leq P$ *then* $I_0 = I_0 \cup \{i\}$;
            *if* $j = n$ *then* $j = 1$;
            *else* $j = j + 1$;
        *while* $|J_0 \cup J_1| < n/25$ *and* $j \neq n_s$;
        *if* $J_0 \cap J_1 \neq \emptyset$ or $\sum_{i\in I_1} w_i > C$ or $\sum_{i=1}^{n} v_i - \sum_{i\in I_0} v_i < R$ *then* break;
        *if* $J_0 \cup J_1 = \emptyset$ *then* $P'$ = EXPCORE($\boldsymbol{x}$, $C$, $R$, $\tilde{\lambda}$, $\tilde{\mu}$ );
        *else*
            *for* $k = 1$ to $n$ *do*
                $C' = C, R' = R$;
                *if* $k \notin J_0 \cup J_1$ *then* $x' = x' \cup \{k\}$;
                *else*
                    *if* $k \in J_1$ *then* $C' = C' - w_j$ and $R' = R' - v_j$;
                    *if* $k < j$ *then* $j = j - 1$;
            *end for*
            $P'$ = REDUCE($x'$, $C'$, $R'$, $j$);
        *end if*
        return $\max(P, P')$;
    *end if*
*end*
```

The pseudocode for the MERGE procedure follows. As input, $N'(s,t)$ is storage for the half of the nodes in the current level of the tree (those nodes which are equal to the unfathomed parent nodes), $N''(s,t)$ is storage for the other half, and $N(s,t)$ is empty storage. Note that $N(s,t)$ will store all unfathomed nodes at the procedure termination and s and t indicate the current depth of the tree. The goal of merge is to select the node from either list which when added to the current end of $N(s,t)$ will preserve the proper ordering. Prior to this addition, the node must pass five fathoming criteria. The sole output is the sorted list $N(s,t)$ of unfathomed nodes for that level of the tree.

```
**procedure** MERGE$\left(N'(s,t), N''(s,t), N(s,t)\right)$
    $k' = 0, k'' = 0, N(s,t) = \emptyset;$
    *while* $k' \leq |N'(s,t)|$ *and* $k'' \leq |N''(s,t)|$ *do*
        select either $k'^{th}$ item from $N'(s,t)$ or $k''^{th}$ item from $N''(s,t)$ such that if the
                node is added to the end of $N(s,t)$, it will satisfy ordering (4-36);
        let the selected node be referred to as $k$ and increase $k'$ or $k''$ as appropriate;
        $ind = 0;$
        *if* $U^k \leq LB$ *then* $ind = 1;$
        *if* $ind \neq 1$ *then*
            *for all* $j \in N(s,t)$ test if $j$ dominates $k$ and let $ind = 1$ if true;
        *if* $ind \neq 1$ *then*
            *if* $W^k - \sum_{i=1}^{s-1} w_i > C$ or $V^k + \sum_{i=t+1}^{n} v_i < R$ *then* $ind = 1;$
        *if* $ind \neq 1$ *then*
            Calculate $U^k = CLU_R(\tilde{\lambda})$, let $\lambda^k = \tilde{\lambda}$ and *if* $U^k \leq LB$ *then* $ind = 1;$
        *if* $ind \neq 1$ *then*
            Calculate $U^k =$
            $\min\left(\left\lfloor z\left(L_R(DKP,\lambda^j)\right)\right\rfloor, \left\lfloor z\left(L_C(DKP,\mu^j)\right)\right\rfloor, z\left(S(DKP,\lambda^j,\mu^j)\right)\right)$
            *if* $U^k \leq LB$ *then* $ind = 1;$
        *if* $ind = 0$ *then*
            $N(s,t) = N(s,t) \cup \{k\};$
            *if* $P^k > LB$ *and* $W^k \leq C$ *and* $V^k \geq R$ *then* update $P;$
        *end if*
    *end while*
```