An Exact Optimization Approach for Relay Node Location in Wireless Sensor Networks

by

Vishal Sairam Jaitra Surendran

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2019 by the
Graduate Supervisory Committee:

Jorge Sefair, Chair
Anthony Grubesic
Pitu Mirchandani

ARIZONA STATE UNIVERSITY

May 2019

ABSTRACT

I study the problem of locating Relay nodes (RN) to improve the connectivity of a set of already deployed sensor nodes (SN) in a Wireless Sensor Network (WSN). This is known as the Relay Node Placement Problem (RNPP). In this problem, one or more nodes called Base Stations (BS) serve as the collection point of all the information captured by SNs. SNs have limited transmission range and hence signals are transmitted from the SNs to the BS through multi-hop routing. As a result, the WSN is said to be connected if there exists a path for from each SN to the BS through which signals can be hopped. The communication range of each node is modeled with a disk of known radius such that two nodes are said to communicate if their communication disks overlap. The goal is to locate a given number of RNs anywhere in the continuous space of the WSN to maximize the number of SNs connected (i.e., maximize the network connectivity). To solve this problem, I propose an integer programming based approach that iteratively approximates the Euclidean distance needed to enforce sensor communication. This is achieved through a cutting-plane approach with a polynomial-time separation algorithm that identifies distance violations. I illustrate the use of my algorithm on large-scale instances of up to 75 nodes which can be solved in less than 60 minutes. The proposed method shows solutions times many times faster than an alternative nonlinear formulation.

DEDICATION

To,

My grandmother, Sumathy Aravindan, my parents, Seetharam Surendran and Lalita Surendren, my sister Darshna Surendran and other family members for their constant love, support and encouragement.

ACKNOWLEDGMENTS

I would first like to thank my thesis advisor Dr. Jorge A. Sefair for being a wonderful educator and mentor. I would also like to thank Dr. Pitu Mirchandani and Dr. Anthony Grubesic for being committee members for this thesis and their valuable inputs. Finally, I must express my very profound gratitude to my parents and to my family for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Background

Advances in sensor design, information and wireless technologies have resulted in the rise of Wireless Sensor Networks (WSNs). WSNs collect information on the physical conditions of an environment and transmit wirelessly from sensor to sensor until the data reaches a Base Station(BS) for processing. Sensors must be strategically dispersed in th surveyed area to capture the information of interest. WSNs have been developing at a rapid pace and new applications for this technology are being discovered constantly. Some of the uses of WSNs are in environmental monitoring (Gao *et al.* (2011)), civil infrastructure protection (Saint (2013)), precision agriculture (Mitralexis and Goumopoulos (2015)), health care (Dishongh and McGrath (2010)), toxic gas detection(Rademacher *et al.* (2015)), fire detection(Hariyawan *et al.* (2013)) and supply chain management(Wang *et al.* (2015)).

WSNs deploy multiple sensors, sometimes thousands or more, in the space of interest. Once deployed, they are required to continue sensing for the lifetime of the network without maintenance. These networks often operate under severe restrictions on factors such as battery life, sensing range, data storage, and size. The energy consumption of these sensors can be divided into two categories ; energy for sensing and energy for transmission of the collected data back to the BS. The energy required for transmission is directly proportional to the square of the distance between the two communicating nodes (Ponnusamy (2014)). This means that the sensor's battery capacity must be very high for it to transmit over longer distances. To mitigate this

problem, WSNs use Relay Nodes (RNs) placed within a certain distance to these sensors to collect their signals and transmit the data back to the BS. Relay nodes do not spend battery sensing data and are solely used for signal transmission. This gives them a higher communication range compared to the sensors. Signals are usually transmitted using multi-hop wireless configurations that use both sensors and relays. This approach decreases the energy dissipation of the sensor nodes, so that the lifetime of the networks is increased. Because the cost of relays typically range from tens to hundreds of dollars, minimizing the number of additional relays required to completely connect or increase the connectivity of a sensor network is an important aspect of the WSN design.

In this research, we propose an exact optimization approach for locating a given number of RNs in a multi-hop WSN to maximize the network connectivity. This problem is known as the Relay Node Placement Problem (RNPP) and is proven to be NP-Hard (Suomela (2006)). The key features of the problem studied in this thesis are the following.

- The location of each sensor and base station is known beforehand.
- The communication range of sensors and relays is known. We assume that the communication range is the same across sensors. Relays may have a different range than sensors, but identical between them.
- Relay nodes can be located anywhere in continuous space without any candidate locations.
- A sensor is said to be connected if there is a sequence of sensors and relays that allow information transmission back to the BS using multiple hops. In other words, relay placement must ensure that there exits at least one path for every connected SN to the BS.

- The objective is to locate a given number of relays to maximize the number of connected sensors.

Although we focus on maximizing connectivity, our proposed solution using only one BS can be expanded to calculate the minimum number of relays required to establish full connectivity. Our methods can easily be adapted for WSNs with multiple BSs.

1.2 Related Work

The RNPP problem is a well-studied problem in the literature. The common approach to solve this problem is by using approximation algorithms because of the properties of the subsequent communication graph. The RNPP is related to the Steiner tree problem (STP) (Gondran and Minoux (1984)), which is an NP-Hard problem that has been studied extensively (Robins and Zelikovsky (2000), Arora (1996)). RNPP was studied as The Steiner Tree Problem with Minimum Number of Steiner Points and Bounded Edge Length (STP-MSPBEL) by Lin and Xue (1999). They were the first to show that the problem is NP-Hard and proposed a polynomial time 5-opt approximation algorithm for the problem. Sapre and Mini (2018) provide nature-inspired approximation algorithms to locate RNs. The RNPP problem for connectivity and survivability was studied by Misra *et al.* (2010). They proposed an approximation algorithm for the same while considering the communication range as the edge bound length. A version of the RNPP with a constraint on the maximum number of hops allowed for each signal(delay), called the Delay Constrained RNPP (DCRNPP) was studied by Bhattacharya and Kumar (2010). They presented a local search based greedy heuristic to provide an approximate solution for the problem. The solutions obtained by these methods are not exact and cannot guarantee optimality.

Exact methods to solve the RNPP or any of its versions are less common. In Cheng *et al.* (2008), the authors formulate an optimization problem based on the Steiner Minimum Tree with minimum number of Steiner points to solve the RNPP. The authors in Bari *et al.* (2007) propose an exact method to solve the RNPP using a set of potential positions situated on a hypothetical grid. Nigam and Agarwal (2014) propose an exact method to solve the DCRNPP using candidate locations for the relays. These solution methods reduce the size of the problem by limiting the locations of the relays to pre-determined sites, ignoring the possibility of locating relays in the continuous space containing the WSN. A non-linear formulation for the RNPP in continuous space (RNPPCS) is given in Zhou *et al.* (2018) where the author uses a mathematical model with Euclidean norm to solve the problem exactly. The results from this formulation are used to measure the performance of a suggested algorithm. The instance sizes solvable by this formulation are very limited due to the non-linear nature of the constraints. The formulation works for WSNs containing upto 12 sensors and 5 relays. To the best of our knowledge, there is no exact algorithm in the literature that solves the RNPPCS for reasonable size instances in polynomial time.

This paper provides a mixed-integer programming based polynomial-time cutting-plane algorithm that iteratively approximates the Euclidean distance needed to enforce sensor communication in order to solve the RNPPCS. We also propose several enhancements to the formulation to improve the algorithm's performance. With the help of computational experiments on several randomly generated test instances of various features, we demonstrate that the proposed algorithm is able to optimally solve problem instances of size up to 75 sensors and 50 relays within reasonable CPU time.

The remainder of the paper is organized as follows. In Chapter 2, we present the approximation technique used to linearize the Euclidean distance between nodes for use in our formulation. In Chapter 3, we introduce notation and provide the mixed-integer mathematical formulation for RNPPCS. In Chapter 4, we describe the algorithm used to solve the RNPPCS and discuss and analyze our experimental results in Chapter 5. Finally, we present our final remarks and some directions for future work in Chapter 6.

Chapter 2

PRELIMINARIES

For continuous location problems, the most widely used distance measure is the Euclidean, denoted by $\|\cdot\|$. For any two locations with coordinates $\mathbf{x}$ and $\mathbf{y}$, the Euclidean distance between them is given by $\|\mathbf{x}-\mathbf{y}\|$. In the context of RNPPCS, we use the Euclidean distance to model the communication between sensors. For instance, two sensors with coordinates $\mathbf{x}_1$ and $\mathbf{x}_2$, communicate with each other if $\|\mathbf{x}_1-\mathbf{x}_2\| \leq 2r$ where r is the transmission radius of each sensor. Since the Euclidean distance function is quadratic, it renders the problem non-linear. Mixed-integer non-linear problems often times do not scale well in problem sizes when compared to linear problems. To overcome the non-linear complexity of RNPPCS, we solve it by linearizing the Euclidean distance inequalities using the method suggested in Camino *et al.* (2016). This linearization returns an approximate distance between any two nodes which is used to measure connectivity in the network. Although this approximation by itself would merely produce sub-optimal solutions, our proposed solution uses successive approximations to obtain a solution that satisfies the Euclidean distance constraint.

2.1 Euclidean Norm Linearization

We define n^u vectors in the Euclidean plane whose angles with respect to a horizontal line range in the continuous domain $[0, 2\pi]$. For a given $n^u \in \mathbb{N}$ such that $n^u \geq 3$, and for all $l \in \{1, ..., n^u\} = U$, we denote the x and y coordinates of $\mathbf{v}_l$ vector by

$$\begin{pmatrix} x_l^u \\ y_l^u \end{pmatrix} = \begin{pmatrix} \cos\left(\dfrac{2(l-1)\pi}{n^u}\right) \\ sin\left(\dfrac{2(l-1)\pi}{n^u}\right) \end{pmatrix} \in \mathbb{R}^2 \tag{1}$$

Given the definition in (1), we have that $\|\mathbf{v}_l\| = 1$, $\forall l \in U$ (2). These n^u unit vectors provide a regular discretization of the unit circle with each n^u vectors having an angle $\dfrac{2\pi}{n^u}$ with its neighboring vectors. An example with $n^u = 8$ is given in Figure 2.1.

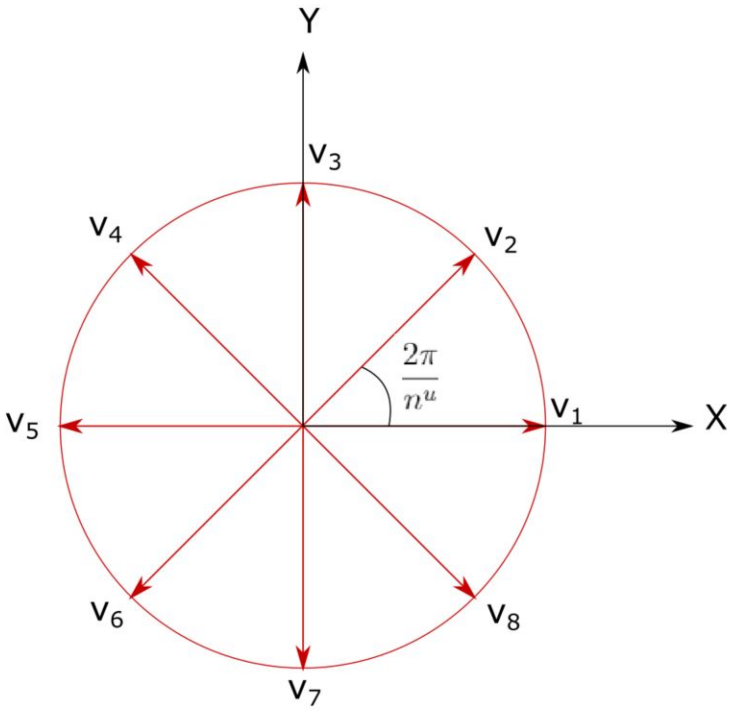

Figure 2.1: Linearization of the Euclidean Plane Using Unit Vectors

Let $\langle \cdot \mid \cdot \rangle$ denote the inner product. For RNPPCS, two nodes can communicate only if they are within a certain distance from each other. Consider two arbitrary locations given by the vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. From Camino *et al.* (2016), we have that

$$\langle \mathbf{x}_1 - \mathbf{x}_2 \mid \mathbf{v}_l \rangle \leq d, \;\; \forall l \in U \Rightarrow \|\mathbf{x}_1 - \mathbf{x}_2\| \leq \frac{d}{\cos\left(\pi/n_u\right)} \tag{3}$$

As n_u increases, the difference $\left| d - \frac{d}{\cos(\pi/n_u)} \right|$ reduces and the approximation tends to the actual Euclidean distance between the two locations. The value of n^u should be large enough such that the distance approximation is reasonably accurate, but it should not be arbitrarily large since it will require a large number of conditions in (3). In Figure 2.2, the regular n^u-sided polygon P represents the feasible region of all vectors whose approximated length is less than or equal to one unit from the origin.

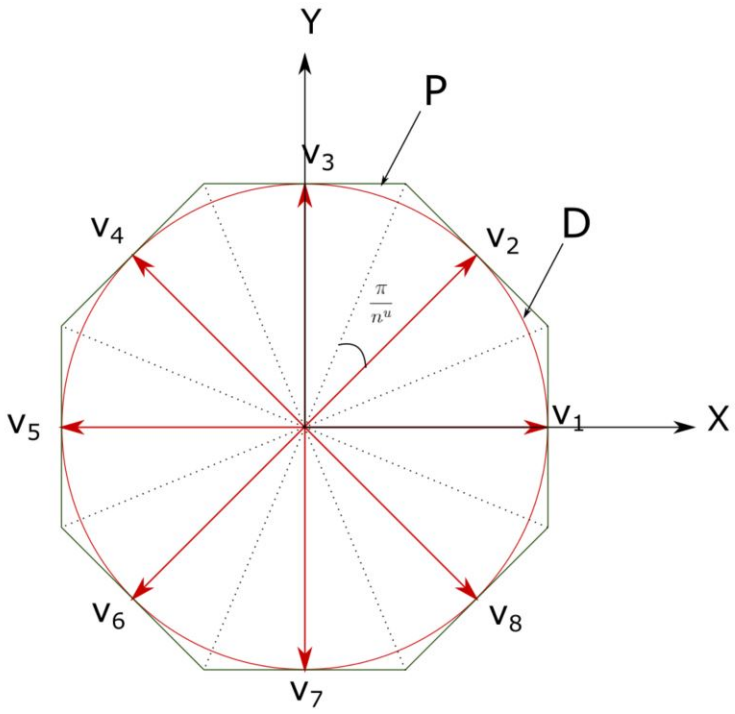

Figure 2.2: Feasible Region Given by the Regular Polygon P

Chapter 3

MATHEMATICAL PROGRAMMING FORMULATION

In this chapter we describe a mixed-integer linear programming formulation for RNPPCS. In our formulation, we have a set S for SNs. The BS is considered as node 1 in S. The x and y coordinates of each element in S are inputted as known parameters x_i^s and y_i^s, $\forall i \in S$. Parameter n is the number of available RNs which allows us to define the set of relay nodes as $K = \{1, ..., n\}$. Set U is a set of n^u distinct vectors such that $\|\mathbf{u}_l\| = 1, \forall l \in U$, whose x and y components are given by x_l^u and x_l^u , $\forall l \in U$. The unit vectors are pre-calculated based on the results from chapter 2, for a given value of n^u. The communication ranges for SNs and RNs are given by r_s and r_k , $\forall s \in S$ and $k \in K$, respectively. The range denotes the maximum distance an SN or RN can communicate with any other node. The range essentially describes a circle with center at the location of each SN and RN and radius r_s and r_k, respectively. Figure 3.1 shows an example for the communication range for a sensor where the red diamond($\diamond$) denotes the sensor's location.

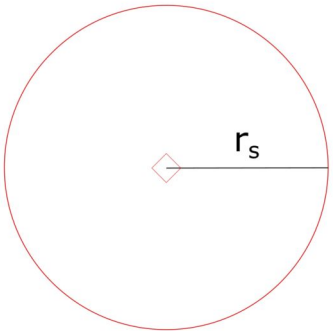

Figure 3.1: Communication Range of a Sensor

Two nodes are capable of communicating signals if their communication circles overlap as shown in Figure 3.2 where the black star ($\star$) denotes the location of a relay node.

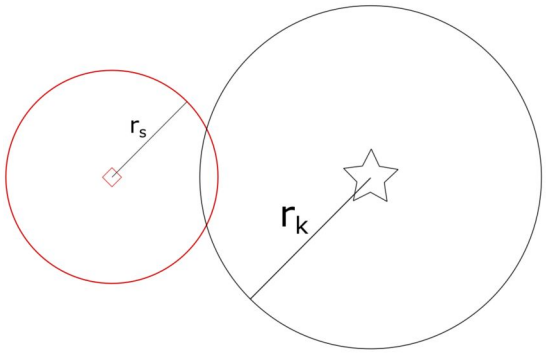

Figure 3.2: Example for Communicating Nodes

Binary parameter p_{st} equals 1 if SNs s and $t \in S$, $s \neq t$, are within communication range of each other. Parameters $\underline{x}$ and $\overline{x}$ are the pre-calculated lower and upper bounds for the x coordinates of the relays. Likewise, $\underline{y}$ and $\overline{y}$ are the bounds on the y coordinates of the relays. We also denote a parameter with a large positive value by $M = \max_{i,j \in S} \|\mathbf{x}_i - \mathbf{x}_j\|$.

We employ five different sets of decision variables in RNPPCS. Variables x_k^r and y_k^r are the x and y coordinates for relay $k \in K$. They are continuous and bounded by their pre-calculated bounds. To model communication between SNs and RNs, we use the binary variable w_{ik}, which is equal to 1 if sensor i communicates with relay k. For communication between RNs, we use binary variable z_{kl} which equals 1 if relays k and l communicate, for $l > k$. Connectivity is captured by the binary decision variable q_s for each $s \in S$. If a path between s and the BS exists, then q_s is equal to

1, and is 0 otherwise. We use continuous variables λ_{sk} to express the location of relay $k \in K$ as the convex combination of sensors $s \in S$. To enforce connectivity, we use a directed graph $G = (N, A)$ where $N = S \cup K$ and $A = \{(i, j)|i, j \in N, i \neq j\}$. We use a continuous variable f_{ij} to capture flow between nodes i and j. Flow only exists between two nodes if they are within communication range of each other. Table 3.1 summarizes the notation of RNPPCS.

Table 3.1: Notation

Category	Notation	Definition
Set	S	Set of all SNs $\cup$ BS
	P	Set of SNs
	K	Set of RNs
	U	Set of unit vectors for distance approximation
Parameter	n_s	Number of sensors including the base station
	n_k	Number of relay nodes
	n^u	Number of unit vectors
	r_k	Communication range (radius) of relays
	r_s	Communication range (radius) of sensors
	x_i^s	x coordinate of sensor $i \in S$
	y_i^s	y coordinate of sensor $i \in S$
	a_{ij}	Euclidean distance between node i and j, $\forall i, j \in N$
	x_l^u	x-component of unit vector $l \in U$
	y_l^u	y-component of unit vector $l \in U$
	p_{st}	pre-processed binary parameter, 1 if sensor s and t are

Table 3.1 continued from previous page

Category	Notation	Definition
		within communication range of each other, 0 otherwise
	$\underline{x}$	minimum value of all x coordinates of sensors
	$\overline{x}$	maximum value of all x coordinates of sensors
	$\underline{y}$	minimum value of all y coordinates of sensors
	$\overline{y}$	maximum value of all y coordinates of sensors
Variable	x_k^r	x coordinate of relay $k \in K$
	y_k^r	y coordinate of relay $k \in K$
	w_{sk}	binary variable, 1 if sensor $s \in S$ and relay $k \in K$ are
		within communication range of each other, 0 otherwise
	z_{kl}	binary variable, 1 if relay $k \in K$ and $l \in K$ are
		within communication range of each other, 0 otherwise
	q_s	binary variable, 1 if sensor $s \in P$ is connected to the sink,
		0 otherwise
	f_{ij}	flow between node $i \in S \cup K$ and node $j \in N \cup K$
	λ_{ij}	convex multiplier for sensor $i \in S$ and relay $k \in K$

3.1 Model

Using the previous notation, RNPPCS is defined as:

$$\max \sum_{s \in P} q_s \tag{4}$$

Subject to:

$$(x_k^r - x_i^s)\, x_l^u + (y_k^r - y_i^s)\, y_l^u \leq r_k + r_s + M\,(1 - w_{sk})\ ,\ \ \forall l \in U, k \in K, i \in S \tag{5}$$

$$(x_k^r - x_l^r)\, x_l^u + (y_k^r - y_l^r)\, y_l^u \leq 2r_k + M\,(1 - z_{kl})\ ,\ \ \forall l \in U, (k,l) \in K : l > k \tag{6}$$

$$\sum_{s \in P : p_{1s} = 1} f_{1s} + \sum_{k \in K} f_{1k} = \sum_{s \in P} q_s \tag{7}$$

$$\sum_{t \in P : p_{st} = 1} f_{st} + \sum_{k \in K} f_{sk} - \sum_{t \in P : p_{ts} = 1} f_{ts} - \sum_{k \in K} f_{ks} = -q_s\ ,\ \ \forall s \in P \tag{8}$$

$$\sum_{s \in P} f_{ks} + \sum_{l \in K} f_{kl} - \sum_{s \in S} f_{sk} - \sum_{l \in K} f_{lk} = 0\ ,\ \ \forall k \in K \tag{9}$$

$$f_{sk} \leq |P| w_{sk}\ ,\ \ \forall s \in S, k \in K \tag{10}$$

$$f_{ks} \leq |P| w_{sk}\ ,\ \ \forall s \in S, k \in K \tag{11}$$

$$f_{kl} \leq |P| z_{kl}\ ,\ \ \forall (k,l) \in K : l > k \tag{12}$$

$$f_{lk} \leq |P| z_{kl}\ ,\ \ \forall (k,l) \in K : l > k \tag{13}$$

$$f_{st} \leq |P| p_{st}\ ,\ \ \forall s \in S, t \in P \tag{14}$$

$$\underline{x} \leq x_k^r \leq \overline{x}\ ,\ \ \forall k \in K \tag{15}$$

$$\underline{y} \leq y_k^r \leq \overline{y}\ ,\ \ \forall k \in K \tag{16}$$

$$x_k^r, y_k^r \in \mathbb{R}\ ,\ \ \forall k \in K \tag{17}$$

$$w_{sk} \in \{0,1\}\ ,\ \ \forall s \in S, k \in K \tag{18}$$

$$z_{kl} \in \{0,1\}\ ,\ \ \forall (k,l) \in K : l > k \tag{19}$$

$$q_s \in \{0,1\}\ ,\ \ \forall s \in S \tag{20}$$

$$f_{ij} \geq 0\ ,\ \ \forall (i,j) \in A \tag{21}$$

The model uses a maximum flow network structure to connect as many sensors as possible. Equation (4) is the objective function and it aims to maximize the sum of the binary variables q_s. That is, it maximizes the number of SNs connected to BS.

The maximum value for the objective function would be equal to $|P|$. Constraints (5) approximate the distance between each sensor and relay using the inner product. The binary variable w_{sk} is equal to 1 if the two nodes are within $r_k + r_s$ units of distance from each other, and is 0 otherwise. Constraints (6) approximate the distance between each pair or relays using the inner product. The binary variable z_{kl} is equal to 1 if the two relays are within $2r_k$ units of distance from each other, and is 0 otherwise. Constraint (7) is the flow balance constraint for BS where the maximum flow out of the BS is equal to $\sum_{s \in N} q_s$. Constraints (8) are the flow balance constraints for all SNs, which are treated as demand nodes with a demand of $-q_s$. Constraints (9) are the flow balance constraints for RNs. The RNs are treated as transshipment nodes. Constraints (10) and (11) allow flow between sensor s and relay k only if they are within communication range. Constraints (12) and (13) are similar but for relay-relay communication. Constraint (14) enforces the same condition between sensors. Constraints (15) and (16) specify the upper and lower bounds on the coordinates of each relay based on the maximum and minimum values of the sensors. Additionally, (17), (18), (19), (20) and (21) are the variable type constraints.

3.2 Model Enhancements

To enhance the performance of our formulation, we added various constraints that help reduce the feasible region in the linear relaxation and may help converge to optimality quicker. The basis for these constraints lies in the behavior of the linear relaxation of the model in (4) - (21).

3.2.1 Valid Inequalities

$$q_s \leq \sum_{k \in K} w_{sk} \ , \ \ \forall s \in S : \sum_{t \in S} p_{st} = 0 \qquad (22)$$

Constraints (22) imply that any sensor s that is not within communication range

of any other sensor, can only be connected to the network ($q_s = 1$) if it is connected to at least one relay, i.e., $w_{sk} = 1$, exists for at least one $k \in K$

$$w_{sk} + w_{tk} \leq 1 \ , \ \ \forall k \in K, (s,t) \in S : a_{st} > 2(r_k + r_s) \qquad (23)$$

Constraints (23) imply that no two sensors that are at a distance greater than $2(r_k + r_s)$ can be connected to the same relay. This can be seen by geometric construction in Figure 3.3, where The minimum distance between two sensors that allows them to be connected to the same relay is $2(r_k + r_s)$.

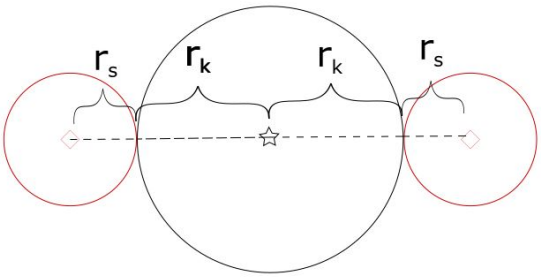

Figure 3.3: Constraint 23 Geometric Construction

3.2.2 Symmetry-breaking Constraints

$$x_k^r \leq x_{k+1}^r \ , \ \ \forall k = 2, ..n_k - 1 \qquad (24)$$

Constraints (24) break the symmetry in the problem. Because RNs are identical, any permutation of the relay labels represent the same feasible solution. Constraints (20) help reduce the number of feasible permutations by imposing a non-decreasing order on the x coordinates.

3.2.3 Convex Hull Constraints

$$x_k^r = \sum_{s \in S} \lambda_{sk} x_s^s \ , \ \ \forall k \in K \qquad (25)$$

$$y_k^r = \sum_{s \in S} \lambda_{sk} y_s^s \ , \ \ \forall k \in K \tag{26}$$

$$\sum_{s \in S} \lambda_{sk} = 1 \ , \ \ \forall k \in K \tag{27}$$

The idea behind introducing these constraints is to limit the search space of the model. The conjecture is that there is at least one optimal solution in which the RNs are located in the convex hull of the sensors including BS. An example is shown in Figure 3.4 and 3.5. Constraints (25) and (26) imply that the relay coordinates must be written as a convex combination of coordinates of those sensors that are the vertices of $conv\{(x_1^s, y_1^s), (x_2^s, y_2^s), ..., (x_{|S|}^s, y_{|S|}^s)\}$. Constraint (27) requires the linear combinations in (25) and (26) to be convex. Constraints (15) and (16) are not used when the convex hull constraints are in place to avoid redundancy.

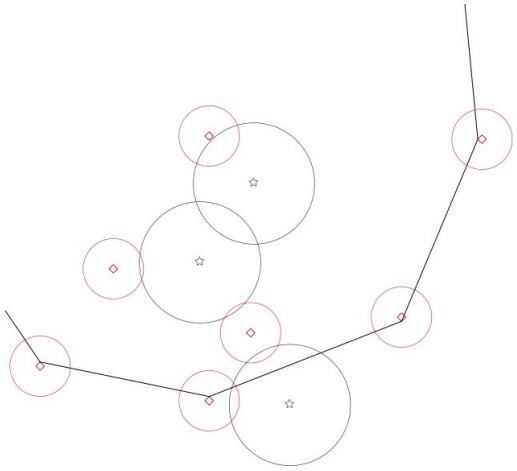

Figure 3.4: Feasible Solution Without Convex Hull Constraints

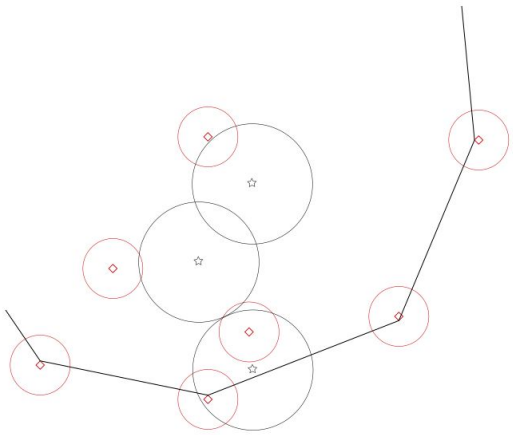

Figure 3.5: Feasible Solution With Convex Hull Constraints

Chapter 4

SOLUTION APPROACH

Our solution approach is a cutting-plane method that iteratively adds constraints (5) and (6) if there is a distance violation. We use Julia 1.0.3 to implement our solution algorithm. In particular, our method relies on the solver callback abilities of Julia to customize the solution process on the run. For example, we can add cutting planes, run custom heuristics to find feasible solutions, and add new constraints only when need as lazy constraints. We exploit this functionality to identify violations and correct them with the appropriately formed constraints using only one branch and bound tree, avoiding the solution of new problems from scratch.

4.1 Algorithm

Algorithm 1 describes our cutting-plane approach. Line 1 indicates the required initializations. Line 2 solves RNPPCS to obtain a solution for $(x_1^r, y_1^r), ..., (x_{|K|}^r, y_{|K|}^r)$. Since the distances are approximated during the solution process using the techniques introduced in Chapter 2, Line 3 denotes the calculations made to identify the true distances between node pairs that communicate using the relay locations obtained from Step 2. Line 4 defines sets Ω (for sensor-relay pairs) and Γ (for relay-relay pairs) that contain node pairs that violate the maximum distance allowed for communication. Line 5 denotes the start of a while loop that adds constraints for violating pairs. If sets Ω and Γ are empty which means there are no violations found for the true distances, the solution is declared optimal and the optimal locations and objective function value are extracted. If there are any violations, the loop executes.

Line 6 initiates a for loop for all violating sensor-relay pairs. The x component of the new unit vector for i and j that makes the current solution infeasible are calculated from the formula in Line 7. Similarly, the y component is calculated in Line 8. The new constraint for i and j is added to the model in Line 9. Since we calculate the unit vector in only one orientation for every i,j pair in each iteration, we do not account for the case where the reverse orientation is required to break the feasibility of the current solution. Line 10 adds a similar constraint using the unit vector in the opposite direction to counter this. Likewise, for every violating pair, two new constraints are added per loop iteration. Line 11 denotes the termination of the For loop for sensor-relay pairs. At Line 12, a second for loop is initiated for all communicating relay-relay pairs that violate the distance requirements. The x and y components of the required unit vectors are generated similar to Line 7 and 8. In Lines 15 and 16, the constraints required to make the distance for the current relay-relay pair infeasible are added. Line 17 indicates the end of the second for loop. In Line 18, the model arrives at a new solution. The new Euclidean distances are calculated in Line 19 in similar manner to Line 3. If no further violations are found, the while loop terminates. Line 15 indicates the optimal solution and objective value being returned by the model.

Algorithm 1: Cutting Plane Algorithm for RNPPCS

Pre-calculate p_{st} and generate the valid inequalities;

Solve RNPPCS and obtain a solution $(x^r_1, y^r_1), (x^r_2, y^r_2), ..., (x^r_{|K|}, y^r_{|K|})$;

Calculate the euclidean distances: a_{ij}, $\forall i, j \in A$, using the locations obtained in Step 2;

Calculate $\Omega = \{(s,k) : s \in S, k \in K, w_{sk} = 1, a_{sk} > r_s + r_k + \epsilon\}$ and $\Gamma = \{(k,l) : k, l \in K, l > k, z_{kl} = 1, a_{kl} > 2r_k + \epsilon\}$;

while $\Omega \neq \emptyset$ *and* $\Gamma \neq \emptyset$ **do**

  forall $i, j \in \Omega$ **do**

    $\hat{x}^u = \dfrac{x^s_i - x^k_j}{a_{ij}}$;

    $\hat{y}^u = \dfrac{y^s_i - y^k_j}{a_{ij}}$;

    Add $(x^k_j - x^s_i)\hat{x}^u + (y^k_j - y^s_i)\hat{y}^u \leq r_k + r_s + M(1 - w_{ij})$ to model;

    Add $(x^k_j - x^s_i)(-\hat{x}^u) + (y^k_j - y^s_i)(-\hat{y}^u) \leq r_k + r_s + M(1 - w_{ij})$ to model;

  end

  forall $i, j \in \Gamma$ **do**

    $\hat{x}^u = \dfrac{x^r_i - x^r_j}{a_{ij}}$;

    $\hat{y}^u = \dfrac{y^r_i - y^r_j}{a_{ij}}$;

    Add $(x^r_j - x^r_i)\hat{x}^u + (y^r_j - y^r_i)\hat{y}^u \leq 2r_k + M(1 - z_{ij})$ to model;

    Add $(x^r_j - x^r_i)(-\hat{x}^u) + (y^r_j - y^r_i)(-\hat{y}^u) \leq 2r_k + M(1 - z_{ij})$ to model;

  end

  Solve RNPP and obtain a solution $(x^r_1, y^r_1), (x^r_2, y^r_2), ..., (x^r_{|K|}, y^r_{|K|})$;

  Calculate the euclidean distances: a_{ij}, $\forall i, j \in A : w_{ij} = 1$, using the locations obtained in Step 2;

end

Return optimal $(x^r_1, y^r_1), (x^r_2, y^r_2), ..., (x^r_{|K|}, y^r_{|K|})$ and $\sum_{s \in N} q_s$

Algorithm 1 terminates in a finite number of steps because only a finite number of unit vectors are needed to approximate the Euclidean distance with ϵ-precision

Chapter 5

COMPUTATIONAL RESULTS

In this chapter, we examine the performance of our formulation by solving the RNPPCS for a set of randomly generated instances. In Section 5.1 we describe our instance generation procedure and in Section 5.2 we illustrate optimal solutions for a few of them. In Section 5.3, we present our results for a broader set of instances of various sizes. To perform our computations, we use Julia with Gurobi 8.1 as optimization solver on a laptop computer with an Intel Core i7 2.40 GHz processor and 16.0 GB RAM.

5.1 Instance Generation

We create instances ranging from 15 SNs to 75 SNs deployed within a region of area 200 times 200 units. For each instance with a given number of sensors, we use two different relay counts. The first combination has as many relays as one third the number of sensors. The second combination has as many relays as two thirds the number sensors. For example, we test an instance size of 30 sensors with 10 and 20 relays. For each sensor-relay combination, we generate five random instances (replications). The communication range of the SNs was considered to be 10 units and for the RNs it was 20 units. In all instances, the BS was considered to be in the center, i.e. at coordinate (100,100). The instances were generated with all the SNs lying in the first quadrant of the coordinate axes however this is not a requirement since the formulation is general enough to work with negative values for coordinates of any node. The coordinates of the SNs were generated randomly using a uniform distribution between 0 and 200. This was done to closely mimic some of the WSN

designs in literature. Our formulation is easily adaptable to multiple Base Stations. Three different models were tested. The first model tested (Model A) did not contain valid inequalities or any other performance enhancing constraints. The second model (Model B) featured all valid inequalities stated in chapter 3 but does not feature the convex hull constraints. The third model (Model C) is the full model with all valid inequalities as well as convex hull type constraints.

5.2 Illustrative Example

In this section , we describe in detail the features of a RNPPCS's optimal solution. The communication range of the SNs (r_s) is 10 and the same for RNs (r_k) is 20. In Figures 5.1 - 5.4, the green triangle ($\triangle$) represents the BS location, the red diamonds ($\diamond$) denote sensor locations and the black stars ($\star$) represent optimal RN locations. An instance of 30 SNs is shown in Figure 5.1. The optimal solution obtained using models A, B and C are shown in Figures 5.2 - 5.4. The three models all produced different optimal locations. This is normal since there are multiple RN location configurations that can connect the same number of sensors.

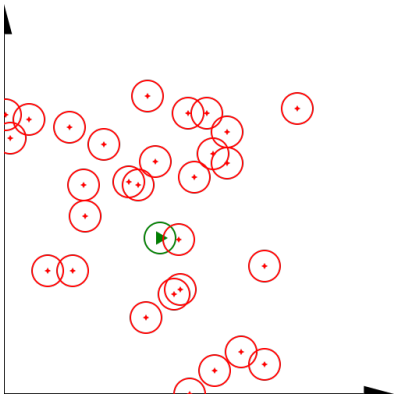

Figure 5.1: Sensor Locations for $|S| = 30$

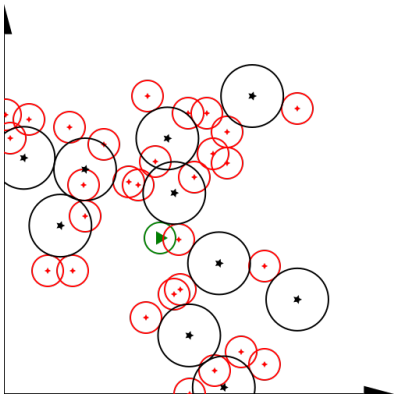

Figure 5.2: Model A Optimal Solution for $|K| = 10$

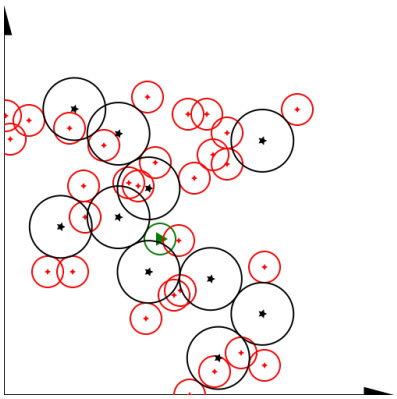

Figure 5.3: Model B Optimal Solution for $|K| = 10$

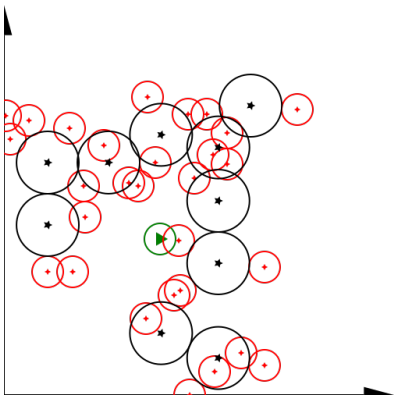

Figure 5.4: Model C Optimal Solution for $|K| = 10$

5.3 Results

In this section, we compare the performance of the different proposed models. The first three columns of Table 5.1 describe the type of instance solved, including the number of SNs ($|S|$), number of RNs ($|K|$) and instance ID. The last three columns contain the run time in seconds of the different models. A value of "-" indicates that the model did not solve in 7200 seconds. In general, as the number of sensors increases, the difficulty of the problem also increases which makes the solve times longer. This behavior is not consistent for certain instances due to unique characteristics such as the sensors being spread out too far within the WSN (denoted as sparse network) and the number of RNs available are not enough to connect all of them. Further study of these behaviors can give more insight into improving model performance.

Table 5.1: Computational Results

S	K	Replication	Model A	Model B	Model C
15	5	1	50.11	11.25	10.11
		2	82.06	5.1	3.08
		3	67.34	3.69	4.21
		4	63.44	3.72	4.38
		5	46.99	3.7	5.91
	10	1	-	756.41	289.92
		2	10.63	4.72	3.1
		3	9.17	9.89	2.86
		4	3.58	10.03	3.48
		5	48.98	44.49	38.86
30	10	1	15.61	19.2	11.7
		2	-	-	2099.65
		3	14.02	7.13	5.1
		4	48.98	47.39	117.72
		5	-	-	5902.22
	20	1	30.33	19.7	13.08
		2	31.38	9.3	44.37
		3	44.67	65.76	17.22
		4	35.6	42.84	42.99
		5	34.78	41.26	40.58
50	15	1	42.9	29.69	39.77
		2	36.28	18.96	34.9

Table 5.1 continued from previous page

S	K	Replication	Model A	Model B	Model C
		3	62.87	18.05	34.8
		4	47.53	14.8	26.71
		5	42.89	81.56	42.26
	35	1	72.14	154.22	117.13
		2	111.93	162.31	87.05
		3	196.15	183.94	187.65
		4	212.28	49.13	152.6
		5	197.68	33.94	110.46
75	25	1	278.62	66.9	152.02
		2	156.15	196.91	47.82
		3	181.77	201.12	53.73
		4	195.03	64.31	185.32
		5	170.19	171.19	81.65
	50	1	1083.03	622.04	522.12
		2	186.18	291.16	672.35
		3	1207.28	201.12	475.17
		4	1173.17	662.53	403.73
		5	188.65	407.64	422.24

Chapter 6

FINAL REMARKS AND FUTURE WORK

This thesis presents a framework to solve the RNPPCS exactly. We formulate this problem as a mixed-integer program. To improve running times, we introduce multiple valid inequalities and the concept of locating RNs inside the convex hull of the SNs. Using clever linearizations and an iterative approach for adding constraints, we solve large instances in reasonable time. Our approach can also be easily be adapted for WSNs with multiple BSs as well as three-dimensional WSNs. A future research path is to vary the number of unit vectors used to approximate the locations and identify further model characteristics that can be exploited to decrease the solve times. Additionally, we can adapt our approach to identify the minimum number of RNs required for full connectivity (wherein there exists a path from each SN to the BS). A potential algorithm would be to start with a pre-calculated (based on the WSN's characteristics) estimate for the initial solve run and then implement a binary search algorithm that either increases or decreases the number of RNs at each step to converge to the true minimum number of RNs required.

The combination of the approximation technique and cutting plane algorithm we propose can in fact be adapted for any location problem in continuous space.

REFERENCES

Arora, S., "Polynomial time approximation schemes for euclidean tsp and other geometric problems", in "Proceedings of 37th Conference on Foundations of Computer Science", pp. 2–11 (IEEE, 1996).

Bari, A., A. Jaekel and S. Bandyopadhyay, "Optimal placement of relay nodes in two-tiered, fault tolerant sensor networks", in "2007 12th IEEE Symposium on Computers and Communications", pp. 159–164 (IEEE, 2007).

Bhattacharya, A. and A. Kumar, "Delay constrained optimal relay placement for planned wireless sensor networks", in "2010 IEEE 18th International Workshop on Quality of Service (IWQoS)", pp. 1–9 (IEEE, 2010).

Camino, J.-T., C. Artigues, L. Houssin and S. Mourgues, "Linearization of euclidean norm dependent inequalities applied to multibeam satellites design", (2016).

Cheng, X., D.-Z. Du, L. Wang and B. Xu, "Relay sensor placement in wireless sensor networks", Wireless Networks **14**, 3, 347–355 (2008).

Dishongh, T. J. and M. McGrath, *Wireless sensor networks for healthcare applications* (Artech House, 2010).

Gao, R., H. Zhou and G. Su, "A wireless sensor network environment monitoring system based on tinyos", in "Proceedings of 2011 International Conference on Electronics and Optoelectronics", vol. 1, pp. V1–497 (IEEE, 2011).

Gondran, M. and M. Minoux, *Graphs and algorithms* (Wiley, 1984).

Hariyawan, M., A. Gunawan and E. Putra, "Wireless sensor network for forest fire detection", Telkomnika **11**, 3, 563 (2013).

Lin, G.-H. and G. Xue, "Steiner tree problem with minimum number of steiner points and bounded edge-length", Information Processing Letters **69**, 2, 53–57 (1999).

Misra, S., S. D. Hong, G. Xue and J. Tang, "Constrained relay node placement in wireless sensor networks: Formulation and approximations", IEEE/ACM Transactions on Networking (TON) **18**, 2, 434–447 (2010).

Mitralexis, G. and C. Goumopoulos, "Web based monitoring and irrigation system with energy autonomous wireless sensor network for precision agriculture", in "European Conference on Ambient Intelligence", pp. 361–370 (Springer, 2015).

Nigam, A. and Y. K. Agarwal, "Optimal relay node placement in delay constrained wireless sensor network design", European Journal of Operational Research **233**, 1, 220–233 (2014).

Ponnusamy, V., "Energy analysis in wireless sensor network: a comparison", International Journal of Computer Networks and Communications Security **2**, 9, 328–338 (2014).

Rademacher, S., K. Schmitt and J. Wöllenstein, "Wireless gas sensor network for the spatially resolved measurement of hazardous gases in case of a disaster", Procedia engineering **120**, 310–314 (2015).

Robins, G. and A. Zelikovsky, "Improved steiner tree approximation in graphs.", in "SODA", pp. 770–779 (Citeseer, 2000).

Saint, M., "Cyber-physical systems for critical infrastructure protection: A wireless sensor network application for electric grid monitoring", (2013).

Sapre, S. and S. Mini, "Optimized relay nodes positioning to achieve full connectivity in wireless sensor networks", Wireless Personal Communications pp. 1–20 (2018).

Suomela, J., "Computational complexity of relay placement in sensor networks", in "International Conference on Current Trends in Theory and Practice of Computer Science", pp. 521–529 (Springer, 2006).

Wang, J., H. Wang, J. He, L. Li, M. Shen, X. Tan, H. Min and L. Zheng, "Wireless sensor network for real-time perishable food supply chain management", Computers and Electronics in Agriculture **110**, 196–207 (2015).

Zhou, C., A. Mazumder, A. Das, K. Basu, N. Matin-Moghaddam, S. Mehrani and A. Sen, "Relay node placement under budget constraint", in "Proceedings of the 19th International Conference on Distributed Computing and Networking", p. 35 (ACM, 2018).

APPENDIX A

JULIA CODE FOR RNPPCS

```
using JuMP, Gurobi

# --- data specification --- #

# Parameters:
nk = 50
ns = 75
nu = 8
dr = 20
ds = 10
M = 1000
tolerance = 0

#Sets:
K = collect(1000:1000 + nk - 1)
S = collect(1:ns)
U = collect(1:nu)

#X coordinates:
xs = Dict(
1 => 100,
2 => 152,
3 => 187,
4 => 82,
5 => 95,
6 => 133,
7 => 118,
8 => 177,
9 => 85,
10 => 122,
11 => 70,
12 => 14,
13 => 88,
14 => 63,
15 => 26,
16 => 25,
17 => 61,
18 => 161,
19 => 47,
20 => 76,
21 => 65,
22 => 167,
23 => 34,
24 => 81,
25 => 177,
26 => 113,
27 => 111,
```

```
28 => 29,
29 => 94,
30 => 196,
31 => 133,
32 => 144,
33 => 193,
34 => 139,
35 => 13,
36 => 154,
37 => 137,
38 => 13,
39 => 18,
40 => 56,
41 => 52,
42 => 20,
43 => 67,
44 => 64,
45 => 14,
46 => 148,
47 => 185,
48 => 128,
49 => 54,
50 => 18,
51 => 103,
52 => 110,
53 => 164,
54 => 4,
55 => 49,
56 => 103,
57 => 112,
58 => 58,
59 => 108,
60 => 75,
61 => 100,
62 => 24,
63 => 160,
64 => 182,
65 => 59,
66 => 168,
67 => 173,
68 => 81,
69 => 34,
70 => 32,
71 => 30,
72 => 167,
73 => 96,
74 => 68,
```

```
75 => 104)

#Y coordinates:
ys = Dict(
1 => 100,
2 => 56,
3 => 11,
4 => 123,
5 => 115,
6 => 151,
7 => 132,
8 => 99,
9 => 63,
10 => 52,
11 => 42,
12 => 184,
13 => 4,
14 => 127,
15 => 172,
16 => 88,
17 => 130,
18 => 105,
19 => 175,
20 => 126,
21 => 68,
22 => 125,
23 => 176,
24 => 70,
25 => 39,
26 => 134,
27 => 158,
28 => 74,
29 => 138,
30 => 19,
31 => 154,
32 => 195,
33 => 157,
34 => 146,
35 => 100,
36 => 90,
37 => 9,
38 => 197,
39 => 111,
40 => 114,
41 => 5,
42 => 157,
43 => 186,
```

```
44 => 11,
45 => 11,
46 => 154,
47 => 102,
48 => 171,
49 => 186,
50 => 57,
51 => 167,
52 => 136,
53 => 87,
54 => 11,
55 => 128,
56 => 125,
57 => 163,
58 => 124,
59 => 169,
60 => 24,
61 => 96,
62 => 112,
63 => 80,
64 => 44,
65 => 124,
66 => 165,
67 => 38,
68 => 178,
69 => 191,
70 => 155,
71 => 166,
72 => 129,
73 => 14,
74 => 82,
75 => 63)

# Boundary Identification

lbx = minimum(values(xs))
ubx = maximum(values(xs))
lby = minimum(values(ys))
uby = maximum(values(ys))

# x component of unit vectors

u1 = Dict(
1 => 1,
2 => 0.707107,
3 => 0,
4 => -0.707107,
```

```
5 => -1,
6 => -0.707107,
7 => 0,
8 => 0.707107)

# y component of unit vectors

u2 = Dict(
1 => 0,
2 => 0.707107,
3 => 1,
4 => 0.707107,
5 => 0,
6 => -0.707107,
7 => -1,
8 => -0.707107)


# --- Pre-processing --- #


# sensor-sensor distance generation ("a" parameter)

a = Dict()
[a[s,t] = ((xs[s] - xs[t])^2 + (ys[s] - ys[t])^2)^0.5  for
    s in S, t in S]

for s in S
    for t in S
        a[t,s] = a[s,t]
    end
end

# Binary parameter 'p' values

p = Dict()
[p[s,t] = 0  for s in S, t in S]
for s in S
    for t in S
        if a[s,t] <= 2 * ds
            p[s,t] = 1
            p[t,s] = 1
            else
            p[s,t] = 0
            p[t,s] = 0
        end
    end
```

```
end

# --- Model declaration --- #

rnlo = Model(solver=GurobiSolver())

# --- variables --- #

@variable(rnlo, lbx <= xk[k in K] <= ubx)
@variable(rnlo, lby <= yk[k in K] <= uby)
@variable(rnlo, wsk[s in S, k in K], Bin)
@variable(rnlo, wkk[k in K, l in K; l>k], Bin)
@variable(rnlo, y[s in S], Bin)
@variable(rnlo, f[i in union(S,K), j in union(S,K); j!=1]
   >=0)
@variable(rnlo, lambda[i in S, j in K] >=0)

# --- objective --- #

@objective(rnlo, Max, sum( y[i] for i = 2:ns))

# --- constraints --- #

@constraint(rnlo, con1[s in S, k in K, u in U], (xk[k] -
   xs[s]) * u1[u] + (yk[k] - ys[s]) * u2[u] <= (dr + ds) +
    M * (1 - wsk[s,k]))

@constraint(rnlo, con2[l in K, k in K, u in U; l>k], (xk[k
   ] - xk[l]) * u1[u] + (yk[k] - yk[l]) * u2[u] <= (2 * dr
   ) + M * (1 - wkk[k,l]))

@constraint(rnlo, con3, sum( f[1,s] for s=2:ns if p[1,s]
   == 1) + sum(f[1,k] for k in K) == sum(y[s] for s=2:ns))

@constraint(rnlo, con4[s in S; s != 1], sum(f[s,t] for t =
    2:ns if p[s,t] == 1) + sum(f[s,k] for k in K) - sum(f[
   t,s] for t in S if p[t,s] == 1) - sum(f[k,s] for k in K
   ) == -y[s])

@constraint(rnlo, con5[k in K], sum(f[k,s] for s = 2:ns) +
    sum(f[k,l] for l in K) - sum(f[s,k] for s in S) - sum(
   f[l,k] for l in K) == 0)

@constraint(rnlo, con6[s in S, k in K], f[s,k] <= (ns - 1)
   * wsk[s,k])
@constraint(rnlo, con7[s in S, k in K; s != 1], f[k,s] <=
   (ns - 1)* wsk[s,k])
```

```
@constraint(rnlo, con8[l in K, k in K; l > k], f[k,l] <= (
   ns - 1)* wkk[k,l])
@constraint(rnlo, con9[l in K, k in K; l > k], f[l,k] <= (
   ns - 1)* wkk[k,l])
@constraint(rnlo, con10[s in S, t in S; t != 1], f[s,t] <=
    (ns - 1)* p[s,t])

# --- Valid inequalities --- #

@constraint(rnlo, con11[s in S; sum(p[s,t] for t in S) ==
   1],
            y[s] <= sum(wsk[s,k] for k in K))

for s in S
    for t in S
        if t!= s
        if a[s,t] > 2 * (dr + ds)
            @constraint(rnlo, [k in K], wsk[s,k] + wsk[t,k
               ] <= 1)
        end
      end
    end
end

for i = 1000:1000+nk-2
    @constraint(rnlo, xk[i] <= xk[i+1])
end

# --- convex-hull constraints --- #

@constraint(rnlo, con12[k in K], xk[k] == sum(lambda[s,k]
   * xs[s] for s in S))
@constraint(rnlo, con13[k in K], yk[k] == sum(lambda[s,k]
   * ys[s] for s in S))
@constraint(rnlo, con14[k in K], sum(lambda[s,k] for s in
   S) == 1)

# --- lazy constraints --- #

function lazyconstraintgenerator(cb)

xkt = getvalue(xk)
ykt = getvalue(yk)
wskt = getvalue(wsk)
wkkt = getvalue(wkk)

as = Dict()
```

```
[as[s,k] = ((xkt[k] - xs[s])^2 + (ykt[k] - ys[s])^2)^0.5
   for s in S, k in K]

for s in S
    for k in K
        as[k,s] = as[s,k]
    end
end

ak = Dict()
[ak[k,l] = ((xkt[k] - xkt[l])^2 + (ykt[k] - ykt[l])^2)^0.5
     for k in K, l in K]

for k in K
    for l in K
        ak[l,k] = ak[k,l]
    end
end

for s in S
    for k in K
        if wskt[s,k] == 1.0
         if as[s,k] > dr + ds + tolerance
           xt = xs[s] - xkt[k]
           yt = ys[s] - ykt[k]
           u1t = xt/as[s,k]
           u2t = yt/as[s,k]
           @lazyconstraint(cb, (xk[k] - xs[s]) * u1t + (yk
              [k] - ys[s]) * u2t <= dr + ds + M * (1 - wsk
              [s,k]))
           @lazyconstraint(cb, (xk[k] - xs[s]) * (-u1t) +
              (yk[k] - ys[s]) * (-u2t) <= dr + ds + M * (1
               - wsk[s,k]))
         end
        end
    end
end

for k in K
    for l in K
        if l>k
          if wkkt[k,l] == 1.0
            if ak[k,l] > 2 * dr + tolerance
              xt = xkt[k] - xkt[l]
              yt = ykt[k] - ykt[l]
              u1t = xt/ak[k,l]
              u2t = yt/ak[k,l]
```

```
                @lazyconstraint(cb, (xk[k] - xk[l]) * u1t +
                   (yk[k] - yk[l]) * u2t <= (2 * dr) + M *
                   (1 - wkk[k,l]))
                @lazyconstraint(cb, (xk[k] - xk[l]) * u1t +
                   (yk[k] - yk[l]) * u2t <= (2 * dr) + M *
                   (1 - wkk[k,l]))
              end
            end
          end
      end
end


end

addlazycallback(rnlo, lazyconstraintgenerator)

#--- solve --- #

solve(rnlo)

# --- end of file --- #
```