EXACT AND HEURISTIC APPROACHES TO SOLVING SENSOR PLACEMENT, ROUTING, AND TRACKING PROBLEMS

By

CHRYSAFIS VOGIATZIS

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2014

I dedicate this dissertation to my parents, Ritsa and Tasos, and my brother, Panagiotis.

ACKNOWLEDGMENTS

I would like to thank my advisor and mentor, Dr. Panos M. Pardalos, for his immense help and support during my Ph.D. studies in the University of Florida.

I would also like to extend a warm thank you to the professors and graduate students in the Department of Industrial and Systems Engineering. It is because of you and your support that I have enjoyed every second of the last 4 and a half years. More specifically, I would like to thank Dr. Geunes for giving me the opportunity to teach, Dr. Smith for his support and acumen, Dr. Richard, and all the rest of the professors in the department.

Of course, I have to acknowledge my friend and collaborator, Jose L. Walteros, for our countless conversations, research ideas, and all the time we have spent together.

Last, but definitely not least, I wouldn't have been able to finish my Ph.D. if it wasn't for the unconditional and constant support of my parents and my brother. Mum, Dad, and Panagiotis, this one is for you. Thank you for everything.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

EXACT AND HEURISTIC APPROACHES TO SOLVING SENSOR PLACEMENT, ROUTING, AND TRACKING PROBLEMS

By

Chrysafis Vogiatzis

August 2014

Chair: Panos M. Pardalos
Major: Industrial and Systems Engineering

In this dissertation, the problem of information tracking, dissemination, and spread in sensor networks is studied. At first, an extensive literature review for sensor networks in transportation and logistics networks is presented, along with the major techniques of data collection and utilization. Then, the study proceeds to the well-known multi-sensor multi-target tracking problem; a new graph partitioning scheme is presented that leads to exact and heuristic approaches for tackling the underlying data association problem. The computational results depict the success of these approaches and show that they are indeed viable alternatives to obtaining optimal and other, high quality solutions in a fast and efficient manner. The study also proceeds to propose extensions to the classical multidimensional assignment problem, from both the integer and graph theoretic viewpoint.

In the last part of the dissertation, the problems of information dissemination and routing are tackled. First, a novel problem that aims to detect highly centralized cohesive subgraphs is investigated. The complexity of the problems is derived, along with mathematical formulations to model the problems at hand. An interesting result that was obtained has to do with the ability to model the problems without having an exponential number of constraints, making the formulations compact and easy to use. Computational results show that, even though the hunt for the largest clique has been ongoing since decades ago, in most practical graphs (Erdos-Renyi, Barabási-Albert, online

social networks) it is much smaller cliques that can spread information the fastest. An extension to these centrality-based formulations is then presented, and (k, l)-Influential Cliques are investigated. Once more, the complexity of the problems is derived, along with a mathematical formulation. Further cuts are investigated that speed up the optimization phase, and a combinatorial branch-and-bound approach is proposed. In the numerical experiments, it is easy to derive that the combinatorial branch-and-bound is always faster, especially after applying pre-processing schemes to the original network.

CHAPTER 1
INTRODUCTION

1.1 Sensors and Optimization

Networks and their theory have been on the forefront of research and practice for decades now, as they entail ingredients from numerous fields (discrete mathematics, computer science, mathematical programing) and can be used to model several applications. These applications can be as diverse as evacuation planning and sensor location to online social network monitoring and assignment problems. For an overview of network applications we refer the interested reader to the seminal work of Phillips and Garcia-Diaz [103] and, more recently, the collection of works presented in Goldengorin et al. [61].

Further, recent technological advancements have been the driving force behind a different ever-growing research field: sensor use to collect, analyze, monitor, and diffuse information and knowledge. In general, a sensor is a device that responds to external events. Its "response" can be different, based on its type and use. However, often sensors are employed to monitor events and collect data. In this work, I will interchangeably refer to sensors as monitors whenever they are used to track information.

These advancements have enabled sensing devices to transmit/store information and perform calculations. This rapid improvement of the technical characteristics of sensors have made them prime candidates for a series of applications, like the mutli-sensor multi-target tracking problem [8, 106], area surveillance, telecommunications, and transportation and logistics [120], among others. For an overview of different sensor applications and the state-of-the-art methods the interested reader is referred to [19].

Herein, the focus will be on a set of sensors that are interconnected. We refer to such a network as a sensor network. Sensor networks are categorized as static or dynamic. In static sensor networks, location and placement are of the essence, since they need to be chosen in advance. Hence, a robust and efficient sensor placement is vital in order to ensure connectivity during the lifespan of the sensors.

Proper network and graph theoretic operations are crucial to guarantee that the quality and lifetime of a sensor network be maximized. In the literature, two paramount notions are coverage and connectivity (robustness). In general, an efficient sensor network design should be both well-connected and maximize area surveillance, even under system failures which can be due to an adversary, sensor faults, or quality of communication service.

In this dissertation, three distinct problems are studied: multi-sensor multi-target tracking problems (data extraction and transformation to knowledge); communication protocols based on centrality metrics (network robustness/design); and influential cardinality-constrained subgraph detection (information dispersion, efficient sensor placement).

The multi-sensor mulit-target tracking problem is a well-known optimization problem where a set of sensors S is deployed to track a set of targets T, resulting in a series of measurements $|S| \cdot |T|$, that are then to be assigned to each other in order to minimize data dissimilarity. In this setting, the multi-sensor multi-target tracking problem is a data association problem, and is often modeled as a Multidimensional Assignment Problem (MAP) [104].

Secondly, communication protocols and data routing has been a well-studied problem in sensor literature for years [4, 48]. Routing in wireless sensor networks usually involves two operations, namely data collection and dissemination. These protocols often use tree-based routing [48]. In such protocols, the use of topological features to design transmission patterns across sensors (nodes) has been proposed in [94]. The insight behind these protocols is that higher structural value is often associated with central sensors, rather than sensors that are found close to the boundary. Further, the importance/relevance of a sensor grows proportionally with its participation in paths [34]. The definition of centrality for nodes is extended to connected clusters of restricted diameter. The reason for doing so is two-fold: first, the restricted diameter ensures that

communication of data and potential re-routing can occur faster, and second, using multiple nodes instead of singletons increases the centrality metric significantly.

Last, data diffusion is an indispensable protocol component, which is often motivated by robustness, scaling and energy efficiency [47]. When all nodes need to receive information/queries from authorized external users (i.e., report any movement), several nodes are fed the query, and then the information is disseminated, using the network structure. Most employed protocols for data diffusion include flooding [69, 70], random walk approaches [23, 115] and information gradient diffusion [32]. The latter takes advantage of domain-specific knowledge. An analysis of the performance of gradient-based routing protocols can be found in [47]. In the approach presented herein, the aim is to detect maximum "influence" connected clusters in sensor networks. Such clusters are important because they can be used to feed information into the system and expect its dispersion to be fast. We start by investigating influential cliques, their complexity along with some special case results, and their mathematical formulations. We then proceed to propose a combinatorial branch and bound and a general randomized adaptive search procedure (GRASP) to solve the problem exactly and heuristically.

This dissertation is outlined as follows. Chapter 1 introduces the problems at hand and briefly offers a background study. Then, in Chapter 2, an overview of sensors in transportation and logistics networks is presented. Chapter 3 introduces novel decomposition schemes for the multidimensional assignment problem. In Chapter 4, I propose centrality-based routing schemes for information dispersion through a sensor network. In that part, I also generalize centrality metrics to connected clusters. Furthermore, Chapter 5 generalizes the notion of degree centrality and introduces the novel problem of detecting (k, l)-influential cliques in a graph. Last, Chapter 6 concludes this dissertation and provides insight in possible future work.

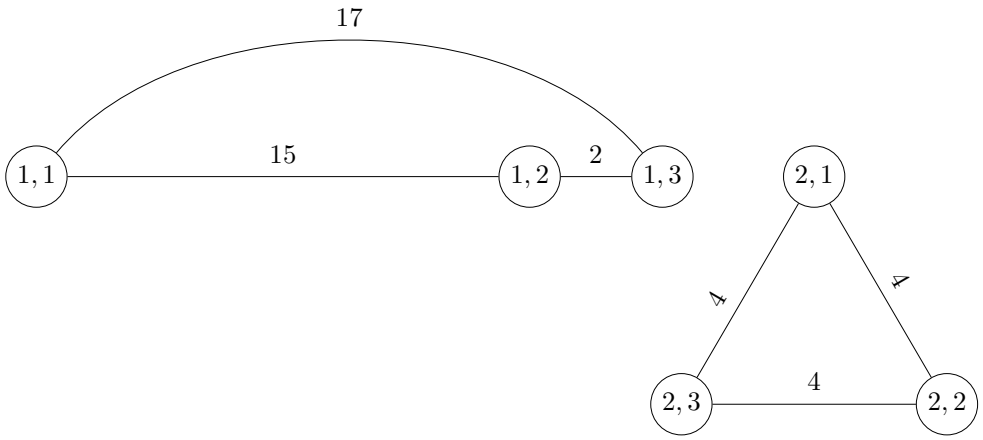


Figure 1-1. Sensor observations and their matching.

1.2 Background

1.2.1 Multi-sensor multi-target tracking problem

Data association is a fundamental optimization problem that aims to match observations with the least dissimilarity. The multi-sensor multi-target tracking problem is a special case, where sensor observations are matched according to their origin [31]. Figure 1-1 offers an example of such sensor observations and their possible matching. In that figure, the nodes in the graph are given by a pair (target, sensor), and hence, target 1 was sensed to be in the position of node (1,1) by sensor 1, in the position of node (1,2) by sensor 2, and so on. The matching presented is called a clique cost matching and is widely used in the multi-sensor multi-target tracking problem framework.

The multi-sensor multi-target tracking problem is usually modeled as a multidimensional assignment problem (MAP) [106]. The MAP can be written as an integer (0-1) program (shown in Formulation 1–1 throughout 1–5).

$$min \quad \sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m} c_{i_1 i_2 \ldots i_n} x_{i_1 i_2 \ldots i_n} \tag{1–1}$$

$$s.t. \quad \sum_{i_2=1}^{m}\sum_{i_3=1}^{m}\cdots\sum_{i_n=1}^{m} x_{i_1 i_2 \ldots i_n} = 1, \qquad \forall i_1 = 1 \ldots m \tag{1–2}$$

$$\sum_{i_1=1}^{m}\cdots\sum_{i_{j-1}=1}^{m}\sum_{i_{j+1}=1}^{m}\cdots\sum_{i_n=1}^{m} x_{i_1 i_2 \ldots i_n} = 1, \qquad \forall i_j = 1 \ldots m, \forall j = 2, \ldots, n-1 \tag{1–3}$$

$$\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_{n-1}=1}^{m} x_{i_1 i_2 \ldots i_n} = 1, \qquad \forall i_n = 1, \ldots, m \tag{1–4}$$

$$x_{i_1 i_2 \ldots i_n} \in \{0, 1\}, \qquad \forall i_1 = 1, \ldots, m, \ldots, i_n = 1, \ldots, m. \tag{1–5}$$

Over the years, a series of methods have been proposed to solve the MAP. Given the inherent $\mathcal{NP}$-hardness of the problem, heuristic methods have attracted most practical interest. Most notable among them are the Lagrangian relaxation proposed by Poore et al. [108], which yielded high quality results, the approximation algorithms of Spieksma [117], GRASP with path relinking [3], and randomized approaches such as the one presented by Oliveira and Pardalos [93].

Herein, we will concentrate on the multidimensional assignment problem with decomposable costs [7] (clique, and star costs namely), where the cost of assignment can be calculated as a function of the pairwise element assignments, i.e., $c_{i_1 i_2 \ldots i_n} = f(c_{i_1 i_2}, \ldots, c_{i_n i_{n-1}})$.

1.2.2 Finding connected subgraphs with maximum centrality

Centrality is a very important property of a node in any graph, since it gives us an understanding of how central that node is. It has been extensively studied in the literature from both a theoretical and experimental standpoint, starting from the seminal papers of Bavelas [15], Leavitt [85], and Sabidussi [114] since the 1950s.

However, centrality has in general been applied to nodes, rather than sets of nodes. In this work, we are extending the definitions to connected clusters. The most used

centrality notions in literature are the following: degree (find a node with maximum degree); closeness (find a node which is closest to every other node); betweenness (find a node that is used by a maximum number of shortest paths); Katz (extension of degree centrality to include path connected nodes); eigenvector (extension of degree centrality to include high-scoring and low-scoring node relations).

More recently, centrality indices have been studied in a survey by [77]. In the same book, Jacob et al. present the state-of-the-art for deriving centrality [71]. As is mentioned therein, computational time is of the essence when computing centrality of a node. However, applying their definition straight to the node might prove to be a naive and inefficient approach.

In [24], the authors show that the betweenness centrality of a node can be computed in $\mathcal{O}(nm + n^2 \log n)$, based on computing a shortest paths tree for every node $i \in V$, which can be precomputed. They also show that a similar calculation leads to the same time bound for closeness centrality in the same paper.

In this work, I proceed to extend the definitions of centrality to cliques, derive the complexity of the problems, and propose mathematical formulations to tackle them.

1.2.3 Influential clique

In this part, we investigate the problem of detecting influential cliques, i.e. cliques that satisfy influence (connectivity) constraints. The problem arises in the context of sensor placement, where a set of sensors needs to be deployed in a network to observe at least k nodes. If, in addition, the sensors are required to communicate with each other (i.e., they need to be placed close enough), then the subset sought on the graph is a clique.

Observe that there are two problems involved with this framework. The first one seeks to maximize the size of the clique subject to an influence constraint, while the latter does the converse: maximizes the number of influenced (adjacent to the clique) nodes, subject to a constraint on the size of the clique. Both problems will be studied herein.

This work is a natural extension of the recent research on the fields of connected maximum coverage, and connected domination. It is closely related to the dominating clique problem [35] on a graph $G(\mathcal{V}, \mathcal{E})$, when we seek a set of nodes $D \subseteq \mathcal{V}$, which form a dominating set and a clique at the same time. In this work, the clique needs not be dominating, so long as it is adjacent to at least k nodes outside the clique. Both the minimum, and the maximum versions of the dominating clique problem are known to be $\mathcal{NP}$-hard optimization problems [79], that can be found efficiently in chordal graphs [78]. An exact algorithm for the minimum dominating clique can be found in [80].

Other related problems stem from the field of connected dominating sets. A recent publication on the field relaxes the clique requirement to a k-club, a connected structure of a diameter that is less than or equal to k [25]. Within the field of sensor placement, research has been focused on detecting robust connected dominating and maximum coverage sets. The interested reader is referred to the works of Zheng et al. [124], Li et al. [86], and Zhou et al. [125].

CHAPTER 2
SENSORS IN TRANSPORTATION AND LOGISTICS NETWORKS

2.1 Sensors in Transportation

In the last decades, the economics of transportation have become of increasing importance for corporations, municipalities and commuters. Due to the large congestion levels every day users have to face, being able to find even an approximate shortest path is vital. However, the analyses that provided us with very useful observations and insight would never be possible if it were not for an automated system of data collection used around the transportation networks.

Since the 1950s, the need for data collection to observe the levels of congestion or the condition of links in the network has appeared, as it can be noted from the Bureau of Transportation Statistics (http://www.rita.dot.gov/bts/sites/rita.dot.gov.bts/files/publications/national_transportation_statistics/index.html). Researchers installed simple or sophisticated sensors in order to measure the volume of the traffic flow on specific links, the average vehicle velocity and the time required to traverse a street among other.

Nowadays, the majority of the vehicles using the network are equipped with routing guidance (usually GPS devices). In addition to that, the increased wireless capacities of cell phones and PDA devices in the last years have made tracking and wireless sensing a common everyday phenomenon. The attempts to combine these increased capabilities with practical transportation problems are going to be described in detail in the following sections.

Moreover, it is important to note that especially in the United States, the appearance of high congestion levels is noticed more often than ever. This comes as a result of the large number of commuters that use the traffic network on an everyday basis [95]. From the reports of the Bureau of Transportation Statistics (http://www.rita.dot.gov/bts/sites/rita.dot.gov.bts/files/publications/national_transportation_

statistics/index.html), it can also be noted that public transportation users have shrunk to insignificant levels, while automobile commuters have never ceased to increase since 1950. It can be seen in the work of O' Toole [95] that the last few years, average vehicle speed is rapidly decreasing, while energy consumption and CO_2 levels are increasing. It becomes evident that it is now more than ever that optimal routing and traffic assignment techniques need to be applied.

The chapter is outlined as follows. In the next section, a presentation of sensor-based methods for collecting useful data for analysis in the fields of transportation will be given. Section 2.3 will then present the most recent advancements in the infamous problems of vehicle routing and traffic assignment. In that section, special attention will be given to modern GPS systems and intelligent vehicle sensors. Last, in section 2.4, the conclusions of the author along with future research possibilities will be presented.

2.2 Data collection and Statistical Analyses

Among the most important and vital ongoing research in any transportation system is the development and implementation of Intelligent Transportation Systems. Systems of this kind can be used to reduce the human factor as far as traffic assignment and vehicle routing are concerned. In addition to that, optimization of traffic flow in intersections will also result in less congestion and accidents [72].

2.2.1 Overview of sensor methods

First of all, let us focus on the tracking methods that have been put to practice in the last decade. These algorithms have extensively investigated computer vision as a means of vehicle tracking. A seminal contribution in the field of traffic monitoring has been presented by Peterfreund [102] in 2002. The author focuses on the snakes active contour models, introduced by Kass, Witkin and Terzopoulos [74] and proposes a stochastic velocity snake model in order to track vehicles in intersections and traffic lights. Equally used is the method of Gardner and Lawton [55] which is common in traffic images.

However, all the above methods are difficult to put into practice when, because of congestion, there is a large number of vehicles at a given intersection at any time. That is the insight behind the Kamijo et al. approach [72] where each vehicle is tracked individually even if in these congestion levels, certain vehicles might be not completely within the sensor/camera range. This phenomenon is referred to as occlusion and has been a major obstacle in traffic monitoring.

2.2.2 Vehicle detection sensors

First of all, let us start by providing the reader with a detailed presentation of the major sensors that are currently available for vehicle detection, as they were noted by Luz and Mimbella [90].

- Inductive Loop: One of the intrusive sensors, most usually installed on the pavement surface. They can also be installed underneath the monitored road by tunneling under the surface of the street. Even though they represent a technology that is greatly understood and very accurate, their installation and maintenance is hard since it requires the disruption of flow for a big time.

- Magnetometer:
  Similar to the inductive loop, a magnetometer is a flexible sensor that can be used for a variety of purposes. Unfortunately, it is prone to erroneous measurements when the traffic is heavy and, hence, it has become unattractive. In addition to that, another drawback that a magnetometer inherits from an inductive loop is that it involves a huge cost of installation and maintenance.

- Active and Passive Infrared:
  Active and passive infrared sensors can be used in combinations in order to cover all measurements of the vehicles on a given road segment. An active infrared sensor can measure accurately the vehicle position and class, while it can also provide us with an estimate on the speed. A passive, on the other hand, can be used for speed measurements. Both categories of infrared sensors share a common disadvantage: they are susceptible to false measurements when visibility is limited (i.e., less than 20 feet) because of weather conditions.

- Ultrasonic:
  Ultrasonic permit multiple lane monitoring at the same time. This is the main advantage that makes them attractive to use. However, they present problematic behavior when functioning under sudden temperature changes and air turbulences.

- Acoustic:

Similarly to ultrasonic sensors, for apparent reasons, acoustic sensors can also monitor more than one lanes at a time with accuracy. Their major problems have to do with cold temperatures and specific traffic patterns. For example they are not recommended for use in large city crossroads where the phenomenon of stop and go traffic is usual.

- Video image processing:
  A video image processing sensor is, in most cases, the ideal candidate. Provides monitoring for as many lanes as the video image capturer can take care of and it is easy to install and maintain without disrupting the normal vehicular flow. A major drawback is the increasing complexity at decoding the sensor measurements into actual data that can be used. Also, as far as requirements for normal installation are concerned, it needs to be ensured that they are installed at a high point, usually 50-60 feet above the monitored road.

2.2.3 Accident detection through image processing

The fundamental idea of Kamijo et al. [72] is the application of a stochastic relaxation algorithm in the detection process. This is vital for the practice, since vehicles have a series of very distinct characteristics, including but not limited to their appearance, their direction and their color. Given that in an intersection, congestion levels are higher and occlusion occurs often, it is impractical to adopt a simple contour method for tracking.

For the modeling purposes of their algorithm, the tracking of a single vehicle is transformed into a labeling method, where each pixel is assigned (i.e., labelled) to a specific vehicle at any time. After all pixels have been labelled, it is part for the deductive method to be applied in order to obtain an initial, but accurate, object mapping.

There are five major components to the deductive process:

- Initialization:
  The background image is set by using a 20 minute image sequence. Also, the entrance points to the intersection are set by defining slits where new intensities of the image (i.e. incoming traffic) appear.

- Generation of new vehicles:
  Whenever along the slits defined, a new intensity is observed, a new ID is assigned to the incoming vehicle. All the pixels sharing that intensity are assigned the same ID.

- Vehicle vector estimation:

At each block of time, the similarity of the vehicle motion is estimated by:

$$(x(t+1), y(t+1)) = (x(t) + u(t), y(t) + v(t)) \tag{2–1}$$

and the motion vector is approximated as the most frequent motion vector of all pixels assigned with the same label,

$$D = \sum_{0 \le d_i \le 8, 0 \le d_j \le 8} |I(i + di + u_i, j + dj + v_j; t + 1) - I(i + di, j + dj; t)|. \tag{2–2}$$

In this case, $I(x, y; t)$ is the intensity of pixel (x, y) at time t.

- Vehicle region update:
  All the vehicle blocks are updated at time $t+1$ compared to time t as per the motion vector obtained in the previous step. If the new intensity difference is smaller than a threshold then the vehicle is considered to be out of the intersection. In the case where the difference is bigger, then it is assumed that a neighboring vehicle is hiding a part of the vehicle tracked.

- Vehicle blocks division:
  At the entrance point of the camera (the slits defined during initialization), simultaneous arrivals may result in characterizing multiple vehicles as one. This is the reason why it is checked always if the motion vectors obtained by a labelled vehicle match. If that is not the case then the pixels corresponding to the same label are divided in order to distinguish between different vehicles.

The last part of this methodology consists of the stochastic relaxation in order to assign certain pixels to more than one vehicles, because of occlusion. The authors then extend the MRF model to include and deal with not only images as in the work by Geman and Geman [57] and by Andrey and Tarroux [5], but also time-axis distribution. Their spatio-temporal MRF model provides an estimation of the current object map based on the information provided in the previous object map along with the current and previous images. After applying the stochastic relaxation procedure to their methodology, the traffic monitoring system for detecting accidents is ready to be examined and tested with the results being successful in a specific intersection, but can be generalized to each topology and geometry [72].

2.2.4 Sensor networks for traffic monitoring

Before the recent advancements in image processing and data extraction, the use of inductive loop detectors was and still remains the most common way to collect information on traffic conditions mainly because of their high levels of accuracy and reliability. However, their installation and their maintenance requires a significant down-time of the arc being updated and hence, they become exceedingly expensive. That is the main reason why in the last years, more sophisticated sensors are being used, such as surveillance cameras, microwave radars, ultrasound and infrared sensors. However, these sensors are less reliable, even though the methods have become more involved over the years, and they also are costly.

Therefore, numerous scientific efforts have been made in order to incorporate cheaper, wireless sensors in the existing infrastructure for statistical and monitoring purposes. The approaches that will be focused upon in this subsection are the ones by Coleri et al. [33, 43].

In the work of Coleri et al. [33], the Traffic-Dot sensor model consists of the following major components:

- processor,
- radio,
- magnetometer,
- battery.

The magnetometer (magnetic sensor) is used for vehicle detection. The power consumption of such a sensor based model is very small, making it an ideal candidate for traffic measurements and data collection for statistical analysis. Its accuracy is also very high, reaching the 97% accuracy that is achieved by inductive loop detectors. In order to increase efficiency and battery utilization the authors have proposed the following linear programming model [43], that is described in Formulation 2–3 throughout 2–9, while the notation is provided in Table 2-1.

Table 2-1. Notation for the model of Coleri et al.

Notation	Definition
$G = (V, E)$	The graph representing the sensor network, with $V = \{1\} \cup V_s \cup V_r$. Node 1 is the access point, $V_s = [2, N]$ are the sensor nodes and $V_r = [N+1, M]$ the relay nodes. If nodes i and j are within transmission range then $(i, j) \in E$.
g_i	Rate of packets per unit time that node $i \in [1, N]$ can transmit.
p_s	Energy spent in sensor when obtaining packets in one packet.
$p_{tx,ij}$	Energy spent for the transmission of a packet from node i to node j per time unit
f_{ij}	Average time required to receive packets at node j from node i.
e_i	The battery energy of each of the parts of the nodes $i \in [1, M]$.

$$\min \sum_{i=1}^{M} e_i \tag{2–3}$$

$$s.t. \sum_j f_{ij} - \sum_j f_{ji} = g_i, \qquad \forall i \in [2, N] \tag{2–4}$$

$$t_d(\sum_j p_{tx,ij} f_{ij} + \sum_j p_{rx} f_{ji} + p_s g_i) \leq e_i, \qquad \forall i \in [2, N] \tag{2–5}$$

$$\sum_j f_{ij} - \sum_j f_{ji} = 0, \qquad \forall i \in [N+1, M] \tag{2–6}$$

$$t_d(\sum_j p_{tx,ij} f_{ij} + \sum_j p_{rx} f_{ji}) \leq e_i, \qquad \forall i \in [N+1, M] \tag{2–7}$$

$$f_{ij} \geq 0, \qquad \forall i, j \in [1, M] \tag{2–8}$$

$$e_i \geq 0, \qquad \forall i \in [1, M]. \tag{2–9}$$

The above linear program has the limitation that it constrains the relay sensors to be in a fixed position in the network. In reality, it is desired to be able to determine the optimal placement of the relay nodes, thus altering dynamically the topology of the network. The resulting formulation is presented in Formulation 2–10 throughout 2–18.

$$\min \sum_{i=1}^{M} e_i \tag{2–10}$$

$$s.t. \sum_j f_{ij} - \sum_j f_{ji} = g_i, \qquad \forall i \in [2, N] \tag{2–11}$$

$$t_d(\sum_j p_{tx,ij} f_{ij} + \sum_j p_{rx} f_{ji} + p_s g_i) \leq e_i, \qquad \forall i \in [2, N] \tag{2–12}$$

$$\sum_j f_{ij} - \sum_j f_{ji} = 0, \qquad \forall i \in [N+1, M] \tag{2–13}$$

$$t_d(\sum_j p_{tx,ij} f_{ij} + \sum_j p_{rx} f_{ji}) \leq e_i, \qquad \forall i \in [N+1, M] \tag{2–14}$$

$$p_{tx,ij} = p_{tx}(d(i,j)), \qquad \forall i, j \in [1, M] \tag{2–15}$$

$$d(i,j)^2 = |l_i - l_j|^2, \qquad \forall i, j \in [1, M] \tag{2–16}$$

$$f_{ij} \geq 0, \qquad \forall i, j \in [1, M] \tag{2–17}$$

$$e_i \geq 0, \qquad \forall i \in [1, M]. \tag{2–18}$$

However, the resulting formulation as can be seen by the reader is a nonlinear, nonconvex problem and hence, the authors propose an approximate algorithm for its solution. Their proposed algorithm is then tested through simulation with remarkable results. Their novel approach in energy management for sensors in a wireless network can result in a more efficient method of data collection in traffic engineering, as can be seen by their work at the Traffic-Dot [33].

2.3 Vehicle Routing and Traffic Assignment

Vehicle routing and traffic assignment problems are two of the most important problems in transportation engineering and planning. Many attempts to solve them in a dynamically changing environment such as the modern urban grid have appeared in literature, however it is in the last few years with the boom of smartphones and sensors that there exist the tools to tackle them successfully.

Online algorithms [73] use recent information as feedback to alter their solutions in order to remain optimal at every instance. For these algorithmic approaches to function effectively reliable information needs to be provided in a fast and efficient way. With the recent increase of guidance devices, such as GPS, and cheap wireless sensors, the possibility to obtain these data appeared.

2.3.1 Sensor-based robotic vehicle routing complexity

Before generalizing the vehicle routing problem to realistic, practical applications involving decision makers in the urban environment, it is important to review the robotic vehicle sensor based routing. This is a common application in automated control systems in industry [26, 39, 111] and, thus, there have been attempts to model and optimize their behavior.

The results of this scientific research has provided insight to researchers as far as online guidance through intelligent automobile systems equipped with sensors are concerned. Usually, in robotic applications, instead of an optimal shortest path for all the vehicles involved, researchers are interested in limiting the selection to a set of available paths and selecting the best among those. This approach has been studied by Inalhan et al. [68] where fixed routes were given to the vehicles and by Gerkey et al. [59], where a fixed roadmap with arcs and nodes was employed.

The above approaches, even though they are practical and present good results, are not theoretically guaranteeing optimality. So, that led Sharma et al. [116] to research the time complexity of this sensor-based vehicle routing problem with no limitations on the selection of routes to the agents utilizing the network. For their work, previous research on communication between robotic systems [75]. The notation for the setup of the problem discussed in [116] is presented in Table 2-2.

The exclusion zone C is a disk of a non constant radius centered at the position of the agent, where there can be no other vehicle. If there is another vehicle at the same time, then a conflict is said to appear and new routing has to occur. The disk is defined in

Table 2-2. Notation for the model of Sharma et al.

Notation	Definition
Q	The square environment where agents are allowed to move of area A.
$(O, D)_i$	The origin-destination pair of agent i.
$t_{0,i}$	The time when an agent is dispatched in the network, $t_{0,i} \geq 0$.
T_i	The time an agent requires in the network until it reaches its destination.
γ_i	The time-dependent path that agent i is following, $\gamma : [0, T_i] \rightarrow Q$.
$v_i(t)$	The velocity of agent i, $v_i(t) \leq v_{max}$.
$C_i(t)$	The exclusion zone C of an agent at time t.

Equation 2–19. As it can be seen, the radius is dependent on the velocity of the agent i at that time t. More specifically, a conflict occurs it there exists a time t_c such that:

- agents i and j are active at time t_c and
- $C_i(t_c) \cap C_j(t_C) \neq \emptyset$.

$$C_i(t) = \{z \in R^2 : ||z - x_i(t)|| \leq r_0 + k||v_i(t)||\} \tag{2–19}$$

The objective of the sensor based vehicle routing problem with robotic agents is to find an optimal routing policy. As such we define a mapping

$$\pi : (O, D) \rightarrow (t_0, T, \gamma)$$

which is safe, that is no conflict occurs at any time. The authors then define T_π to be the time when all agents have safely arrived at their destinations according to policy π. Then, the time complexity is defined as

$$T^*(O, D) = \inf_{\text{safe } \pi} T_\pi(O, D).$$

This formulation leads to interesting theoretical results when it comes to the upper and lower bounds in the time complexity of the problem. There are two cases that were examined:

- Best case scenario.

- Average case scenario.

In the best case scenario, the $(O, D)_i$ pairs are selected so as to minimize the total time required for all agents to remain in the environment. The authors go ahead and prove the following lemma in that case:

Lemma 1. For any set of n (O, D) pairs, such that the average distance between origin and destination points is $\bar{L}$, the time complexity of the problem is $\Omega(\sqrt{n}\bar{L})$.

They also proceed to prove the following lemma for an upper bound on the time complexity of the best case:

Lemma 2. For any $n \in N$, $\exists$ n (O, D) pairs such that the time complexity is $O(\sqrt{n}\bar{L})$, where $\bar{L}$ represents the average distance of any two origin-destination points.

Combining these two lemmas, the authors obtain a very important theorem on the time complexity of the sensor based vehicle routing problem as formulated above. However, the most important aspect of their work is yet to follow with the average case scenario study, where the origin-destination pairs are no longer arbitrarily selected, but are random. In that case, the following lemma is proved by the authors on a lower bound of the time complexity.

Lemma 3. The time complexity of the problem when the set of n origin-destination pairs is randomly selected from the uniform distribution is, with high probability, $\Omega(\sqrt{n})$.

Next, the authors present their algorithmic framework that terminates in $O(\sqrt{n})$ time, hence concluding that with high probability the time complexity of the problem is $O(\sqrt{n})$. Sensor based fully automated vehicles are used for transportation purposes in large facilities [89] or depot centers [119] and, hence, the result of the time complexity of their optimal routing problem is of significant importance in the fields of logistics.

2.3.2 Intelligent vehicle routing through a centralized highway system

The previous results may apply only to supply chain networks and to the optimization of the automated procedures regarding storing, handling and shipping, however inspired a number of researchers in generalizing the notions in the large-scale, real-life traffic system.

The idea of receiving feedback that provides the tripmaker with information on the state of the roads that they are planning to use is not new at all.

Television and radio channels spend time on informing the audiences which streets should be avoided, where the users are experiencing normal traffic flows or if there has been an accident. In the last years, guidance devices, such as the GPS routing system, provide the possibility to obtain traffic data in real-time [91].

As was mentioned in the previous subsection, many attempts to model realistically the problem for real vehicular flows emanate from the robotic world and a fully automated and controlled system of vehicles. That was, also, the insight of Baskar, De Schutter and Hellendoorn, who first proposed an hierarchical traffic control system for intelligent vehicles [10, 12] and then adapted its formulation to obtain a mathematical programming problem in order to come up with the optimal routing of the vehcles [11].

The framework, described in Baskar et al. [10] consists of the following elements:

1. the vehicle controllers, which control the speed and steering of the vehicles by receiving orders from the platoon controllers;
2. the platoon controllers, which take care of the merges and splits of platoons and the vehicle to vehicle distances by receiving control commands from the roadside controllers;
3. the roadside controllers, which are in charge of a segment of the whole network in the infrastructure;
4. and the higher-level controllers, which coordinate the whole network and supervise all other controllers.

Using this infrastructure, the authors focus on optimal routing to each platoon that is currently in the network. The notation that is used by Baskar et al. [11] is given in the Table 2-3.

Table 2-3. Notation used in the autonomous vehicle routing.

Notation	Definition
O	Origin nodes
D	Destination nodes
I	Internal nodes
V	The set of all nodes, $V = O \cup I \cup D$
L	The set of links in the network
(o, d)	One origin-destination pair, $(o, d) \in O \times D$
$L_{o,d}$	The set of links that belong to a route connecting o to d
$D_{o,d}$	The demand of the pair (o, d)
C_l	The capacity of the link $l \in L$
v_l	The speed on link $l \in L$
τ_l	The travel time on link $l \in L$
L_v^{in}	The set of all links incoming to node v
L_v^{out}	The set of all links leaving node v
$x_{l,o,d}$	Decision variables denoting the flows for every pair $(o, d) \in O \times D$
T	The simulation period

The simple linear model that corresponds to the framework described before is given in Formulation 2–20 throughout 2–23.

$$\min \quad J_{links} = \sum_{(o,d)\in O\times D} \sum_{l\in L_{o,d}} x_{l,o,d}\tau_l T \tag{2–20}$$

$$s.t. \quad \sum_{l\in L_o^{out}\cap L_{o,d}} x_{l,o,d} = D_{o,d}, \qquad \forall o \in O, \forall d \in D \tag{2–21}$$

$$\sum_{l\in L_v^{in}\cap L_{o,d}} x_{l,o,d} = \sum_{l\in L_v^{out}\cap L_{o,d}} x_{l,o,d}, \qquad \forall v \in, \forall (o,d) \in O \times D \tag{2–22}$$

$$\sum_{(o,d)\in I_{od,l}} x_{l,o,d} \leq C_l, \qquad \forall l \in L. \tag{2–23}$$

The model is simple to understand since it involves only the flow balance in the network and the capacity constraints for each of the links. The objective function in Equation 2–20 is a measure of the time that the vehicles have to spend while traveling in the network.

In order to more realistically model the problem, the authors then proceed to include queues that can be formed at the entries of the infrastructure. So now the model can be

rewritten as:

$$\min \quad J_{links} + J_{queue} = \sum_{(o,d)\in O\times D} \sum_{l\in L_{o,d}} x_{l,o,d}\tau_l T + \tag{2–24}$$

$$\sum_{(o,d)\in O\times D} \frac{1}{2}(D_{o,d} - F^{out}_{o,d})T^2 \tag{2–25}$$

$$s.t. \quad \sum_{l\in L^{in}_v \cap L_{o,d}} x_{l,o,d} = \sum_{l\in L^{out}_v \cap L_{o,d}} x_{l,o,d}, \qquad \forall v \in, \forall (o,d) \in O\times D \tag{2–26}$$

$$\sum_{(o,d)\in I_{od,l}} x_{l,o,d} \leq C_l, \qquad \forall l \in L \tag{2–27}$$

$$\sum_{l\in L^{out}_o \cap L_{o,d}} x_{l,o,d} \leq D_{o,d}, \qquad \forall o \in O, \forall d \in D \tag{2–28}$$

$$F^{out}_{o,d} = \sum_{l\in L^{out}_o \cap L_{o,d}} x_{l,o,d}. \tag{2–29}$$

J_{queue} is a measure of the time spent by the vehicles in the queues formed in the origin nodes. In order to come up with an estimate for these measures, the authors note that the queue size increases with time with a rate of $D_{o,d} - F^{out}_{o,d}$. Hence, at the end of the simulation the total length is $(D_{o,d} - F^{out}_{o,d})T$ and the average can be calculated as $\frac{1}{2}(D_{o,d} - F^{out}_{o,d})T$. That is how the term of J_{queue} is computed in the objective function above. It is important to note that once more the mathematical program obtained is linear.

Even in this last model, the approach is highly unrealistic. It is not a valid assumption that the demands are static, but have to be considered dynamic in order to accommodate most practical applications. In order to do so, a discretization of the time spent on each link is introduced, with the elementary measurement of T_s. This can be written more clearly as

$$\tau_l = \kappa_l T_s, \text{ where } \kappa_l \in Z^+. \tag{2–30}$$

Letting $q_{o,d}(k)$ be the partial queue length of vehicles traveling from o to d at time k, i.e. $t = kT_s$, and by assuming that the network is initially empty, i.e. $q_{o,d}(k) = 0$ and

$x_{l,o,d}(k) = 0$ for $k \leq 0$ we can now have for each of the origin nodes o:

$$\sum_{l \in L_o^{out} \cap L_{o,d}} x_{l,o,d}(k) \leq D_{o,d}(k) + \frac{q_{o,d}(k)}{T_s} \quad \forall d \in D, \tag{2–31}$$

and by definition $D_{o,d}(k) = 0$ for $k \geq K$. Now, by considering the fact that every vehicle on link l will reach the end of the link after κ_l time segments, we obtain that

$$\sum_{l \in L_v^{in} \cap L_{o,d}} x_{l,o,d}(k - \tau_l) = \sum_{l \in L_v^{out} \cap L_{o,d}} x_{l,o,d}(k), \quad \forall v \in I \quad \forall (o,d) \in O \times D. \tag{2–32}$$

Also for every link, we have the capacity constraints, however taking into consideration the time we are in. So, Constraint 2–27 is now transformed in the dynamic case into

$$\sum_{(o,d) \in I_{od,l}} x_{l,o,d}(k) \leq C_l, \quad \forall l \in L. \tag{2–33}$$

The important part of this modeling approach is the description of the queues formed. The flow is given by a similar constraint to the one presented in Constraint 2–29, which however is transformed in order to accommodate the time factor into

$$F_{o,d}^{out}(k) = \sum_{l \in L_o^{out} \cap L_{o,d}} x_{l,o,d}(k). \tag{2–34}$$

Therefore, the queue length is increasing linearly with the rate of $D_{o,d}(k) - F_{o,d}^{out}(k)$ for the time interval $[kT_s, (k+1)T_s)$ and we get the following equation for the queue length:

$$q_{o,d}(k+1) = \max\left(0, q_{o,d}(k) + (D_{o,d}(k) - F_{o,d}^{out}(k))T_s\right). \tag{2–35}$$

There exist two cases which can be distinguished for the determination of the time $J_{queue,o,d}(k)$ that a vehicle has to spend in the queue formed at an origin o:

- a. The queue length becomes zero while the interval $[kT_s, (k+1)T_s)$.
- b. The queue length remains positive in the same interval.

Let us consider the second case. Defining the time after kT_s at which the queue length becomes zero as

$$T_{o,d}(k) = \frac{q_{o,d}(k)}{F^{out}_{o,d}(k) - D_{o,d}(k)}, \tag{2–36}$$

J_{queue} can now be estimated as

$$J_{queue,o,d}(k) = \begin{cases} \frac{1}{2}(q_{o,d}(k) + Q_{o,d}(k+1))T_s & \text{for the first case} \\ \frac{1}{2}q_{o,d}(k)T_{o,d}(k) & \text{for the second case.} \end{cases} \tag{2–37}$$

In general now, we have

$$J_{queue} = \sum_{k=0}^{K_{end}-1} \sum_{(o,d)\in O\times D} \sum_{l\in L_{o,d}} J_{queue,o,d}(k) \tag{2–38}$$

and

$$J_{links} = \sum_{k=0}^{K_{end}-1} \sum_{(o,d)\in O\times D} \sum_{l\in L_{o,d}} x_{l,o,d}(k)\kappa_l T_s^2. \tag{2–39}$$

So, finally the mathematical formulation becomes

$$\min \quad (J_{links} + J_{queue}) \tag{2–40}$$

$$s.t. \quad \text{Constraints 2–31 throughout 2–35} \tag{2–41}$$

This model is for the second case a nonlinear, nonconvex and nonsmooth problem. As such, this problem is hard to solve and hence, the authors present an approximate solution algorithm. Either way, by transforming the above problem into a mixed integer linear program, there exist several solvers that can solve it efficiently [113]. For this transformation to take place, the properties of Bemporad and Morari [16] can be used:
$[f \leq 0] \iff [\delta = 1]$ is true iff

$$\begin{cases} f \leq M(1-\delta) \\ f \geq \epsilon + (m-\epsilon)\delta, \end{cases}$$

where ϵ is a small positive number. $y = \delta f$ is equivalent to

$$\begin{cases} y \leq M\delta \\ y \geq m\delta \\ y \leq f - m(1-\delta) \\ y \geq f - M\delta. \end{cases}$$

Then, the term of $F_{o,d}^{out}(k)$ from Equation 2–35 can be eliminated, thus

$$q_{o,d}(k+1) = \max\left(0, q_{o,d}(k) + (D_{o,d}(k) - \sum_{l \in L_o^{out} \cap L_{o,d}} x_{l,o,d}(k))T_s\right), \tag{2–42}$$

which still is nonlinear. However, by letting $D_{max,o,d}$ be the maximum demand for $(o,d) \in O \times D$, $F_{max,o,d}$ be the maximum feasible flow (i.e. $F_{max,o,d} = \sum_{l \in L_o^{out} \cap L_{o,d}} C_l$) and $q_{max,o,d}$ be the maximum queue length formed from origin o to destination d and equal to $D_{max,o,d}T_sK_{end}$, then two new parameters can be defined as

$$m_{o,d}^{low} = -F_{max,o,d}T_s \tag{2–43}$$

$$m_{o,d}^{upp} = q_{max,o,d} + D_{max,o,d}T_s, \tag{2–44}$$

hence the following always stands:

$$m_{o,d}^{low} \leq q_{o,d}(k) + (D_{o,d}(k) - \sum_{l \in L_o^{out} \cap L_{o,d}} x_{l,o,d}(k))T_s \leq m_{o,d}^{upp}. \tag{2–45}$$

Now, by introducing the binary variables $\delta_{o,d}(k)$ as

$$\delta_{o,d}(k) = \begin{cases} 1 \text{ iff } q_{o,d}(k) + (D_{o,d}(k) - \sum_{l \in L_o^{out} \cap L_{o,d}} x_{l,o,d}(k))T_s \geq 0 \\ 0 \text{ otherwise.} \end{cases} \tag{2–46}$$

So, applying property 1, Equation 2–42 takes the form of

$$q_{o,d}(k+1) = \delta_{o,d}(k)(q_{o,d}(k) + D_{o,d}(k) - \sum_{l \in L_o^{out} \cap L_{o,d}} x_{l,o,d}(k))T_s). \tag{2–47}$$

2.3.3 Vehicle routing and traffic monitoring using personal sensors

As personal sensors we denote these devices that can provide on the fly feedback and information, such as smartphones, Personal Digital Assistants (PDA) and GPS systems. The main driving idea is that the number of the traffic network users that also owns one of the aforementioned devices has significantly increased in the last years, making the propagation of information easier.

This has been the insight of Thiagarajan et al. [118] in creating a prototype application that gathers the information required through mobile phones and provides online routing based on recent traffic trends. In their work, they present real-time traffic monitoring system tracking the vehicle trajectories using a hidden Markov chain.

In general, three are the major pillars of a high quality online routing algorithm:

1. Accuracy:
   the time estimations and the congestion levels that are used to reroute and guide vehicles in the network have to be close enough to represent the real situation.

2. Energy efficiency:
   it is an important note that a smart phone battery is consumed much faster when an algorithm that requires access to online data is executed continuously. Therefore, a trade-off between the sampling time of the data and a high quality solution needs to be agreed.

3. Time efficiency:
   the algorithm applied needs to be fast and efficient. An algorithm that is too computationally expensive may present optimal routes, however it is unrealistic to assume that it can provide on the fly routes when needed.

However, the above objectives of an online routing algorithm present a number of challenges, the most difficult and common of which are mentioned below:

- Map matching of the trace to the road segment it corresponds to [67, 83].
- Time estimation of specific segments in a route.
- Accuracy is energy consumptive. Sampling GPS has been shown to be [53] far more expensive in terms of energy that WiFi sampling, which unfortunately is less accurate.

For the tracking algorithm proposed, the authors use a hidden Markov chain. By that, they imply that the positions at each sampling period of time are known, however the road segments (i.e., the transitions between positions) are unknown. Given a set of known positions over the time of the vehicle movement, the goal is to detect the maximum likelihood road segments that were used. The algorithm that they propose can be summarized as follows:

- Compute transmission probabilities.
- Compute emission probabilities.
- Employ the Viterbi decoding algorithm.
- Bad zone detection and removal.

In order to compute the transition probabilities, the following notions need to be considered. First of all, there exists a probability that the car will still be in the same road segment for the next sampling period. Also, a car can only change road segments if there exists an intersection between the segment it was on and the segment it is observed to be on. Last, there are limits on the vehicle speeds that prohibit the car to go extremely fast in any given road segment. Mathematically, the notions above are summarized to Constraints 2–48 to 2–50 representing the probability p for a vehicle whose position at sampling time $t-1$ is i while at sampling time t is j:

$$\text{If } i = j, p = \epsilon. \tag{2–48}$$

$$\text{If j and i share no intersection }, p = 0. \tag{2–49}$$

$$\text{If i and j share an intersection }, p = \epsilon \text{ or } p = 0. \tag{2–50}$$

Constraint 2–48 defines the probability that a car is still found in the same road segment and ϵ is defined as

$$\epsilon \leq \frac{1}{dmax + 1}.$$

The third constraint effectively prohibits the vehicle from moving extremely fast at any road segment. If a vehicle is detected at position i at time $t-1$ and then at time t is found at j, then the algorithm computes the time it would normally take the vehicle to traverse this route. If that implies that the car is traveling at a speed that is greater than the threshold speed $S_{outlier}$ defined at 200 mph, then the probability is set to be equal to 0; otherwise it is equal to ϵ.

The next step of the tracking algorithm involves the emission probabilities of the model. The emission probability notion is employed to cover the fact that it is possible for a point to be observed from a road segment that is close but is not necessarily the closest one. So, using a Gaussian function with zero mean N, the emission probability of the road segment i at position l is defined as

$$N(dist(i,l))$$

where $dist(i,l)$ is the Euclidean distance. The variance of N is dependent on the sensor that produced the position and, hence, different variances are used for WiFi and GPS position sampling.

The most important component of the technique applied is the Viterbi decoding algorithm [67] which finds the most likely sequence of hidden states, i.e. road segments, that the vehicle is required to pass through. Last in the sequence, after having obtained a valid route by applying the Viterbi algorithm, the "bad zones" are detected and removed from the route. That way, the authors can ensure that the route is as realistic as possible and they can use it to obtain useful information on the traffic status and the time required to traverse these arcs in real time.

Overall, the algorithm presented was applied to real data, with important results, including the facts that:

- Using WiFi localization, 90% of the routes predicted were within a 10-15% of the optimal route. GPS localization presented optimal results with high accuracy over a sampling period of 30 seconds.

- A hybrid algorithm employing the 30 seconds GPS sample with WiFi localization in between has an improved performance over the two methods mentioned above, however the gains are much less than the energy consumptions.
- Using GPS localization over a sampling period of 20 seconds outperforms the hybrid approach.

2.4 Smartphones and Novel Approaches

The recent boom that smartphones have seen in the last few years has made cheap sensors available to a number of users. Especially when it comes to transportation systems, there is now the possibility of collecting information fast through the wireless networks that support these phones. This insight drove a number of researchers like in [118] to investigate the methods that feedback can be derived from these devices and provided to sophisticated algorithms.

Current research is focusing on sensor-based vehicle routing problems, where vehicles are both providing and provided feedback on the traffic status of their route and the time that is required to traverse links in the network. An important part of the algorithm that would benefit the trip makers' decisions would be the incorporation of historical data of the day or the period in the prediction model. In order to do so, sophisticated time-series approaches [122] and/or kernel regression machine learning are applied to the model. Data mining techniques [66] can also provide us with useful remarks on the traffic behavior throughout a time period.

Another component of these algorithms that needs to be improved significantly is energy consumption. Nowadays, it is known that tracking and routing devices are expensive and use up a significant amount of battery. Therefore, it is not only important to provide travelers with reliable, on-the-fly algorithms for routing, but also algorithms that use up as little energy as possible. These are the major directions for future research that will optimize the procedures of using sensors in routing and traffic assigning. If we were to improve these conditions, then the algorithms discussed in this chapter would certainly be much more accessible to a number of users.

CHAPTER 3
GRAPH PARTITIONS FOR THE MULTIDIMENSIONAL ASSIGNMENT PROBLEM

3.1 Preliminaries

The multi-sensor multi-target tracking problem is a particular case of the data association problem. The problem consists of selecting the most probable of associations among several sensor measurements. Assuming there are n sensors and m measurements, the possible associations are $(m!)^n$ in number, which is intractable in large-scale scenarios. The data association problem specifics may vary depending on the application and assumptions, however it is often formulated as a multidimensional assignment problem.

Before we proceed to give the formulation for the multidimensional assignment problem, it is of interest to show the simple assignment problem version. The Assignment Problem ($2AP$) is a well-known problem in optimization, where two disjoint sets I and J, each containing n elements, are assigned to each other element wise. If elements i and j from the two sets are assigned together, they yield a cost of c_{ij}. A solution is valid, if and only if each element is assigned exactly once. Let variable x_{ij} be defined as follows

$$x_{ij} = \begin{cases} 1, & \text{if element } i \text{ is assigned to element } j \\ 0, & \text{otherwise.} \end{cases}$$

The $2AP$ may now be formulated as in Formulation 3–1 throughout 3–4.

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} \cdot x_{ij} \tag{3–1}$$

$$s.t. \quad \sum_{j=1}^{n} x_{ij} = 1, \qquad \forall\ i = 1, 2, \ldots, n \tag{3–2}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad \forall\ j = 1, 2, \ldots, n \tag{3–3}$$

$$x_{ij} \geq 0, \qquad \forall i, j = 1, 2, \ldots, n, \tag{3–4}$$

where the constraints guarantee that each element is assigned exactly once. Denote the bijective function $\phi : I \rightarrow J$, where $\phi(i)$ is the element from the second set assigned to element i. We can ensure that every element is only assigned once by requiring ϕ to be a permutation vector. This leads to the following permutation formulation, shown in Equations 3–5 throughout 3–7.

$$\min \quad \sum_{i=1}^{n} c_{i\phi(i)} \tag{3–5}$$

$$s.t. \quad \phi(i) \neq \phi(j) \qquad \forall\, j \neq i \tag{3–6}$$

$$\phi(i) \in \{1, 2, \ldots, n\}, \qquad \forall i = 1, 2, \ldots, n. \tag{3–7}$$

The 0-1 integer programming and permutation formulations are equivalent, but they offer different approaches to finding an optimal assignment.

When the number of dimensions in an assignment problem is greater than two, the problem is referred to as a multidimensional assignment problem (MAP), introduced by Pierskala [104]. As an example, consider the problem of assigning jobs, workers, and machines. The MAP is a well–studied problem in literature with numerous applications, including multi–sensor multi–target tracking [17, 107] and data association problems [6, 9]. The MAP is typically formulated as in Formulation 3–8 throughout 3–12.

$$min \quad \sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m} c_{i_1 i_2 \ldots i_n} x_{i_1 i_2 \ldots i_n} \tag{3–8}$$

$$s.t. \quad \sum_{i_2=1}^{m}\sum_{i_3=1}^{m}\cdots\sum_{i_n=1}^{m} x_{i_1 i_2 \ldots i_n} = 1, \qquad \forall i_1 = 1 \ldots m \tag{3–9}$$

$$\sum_{i_1=1}^{m}\cdots\sum_{i_{j-1}=1}^{m}\sum_{i_{j+1}=1}^{m}\cdots\sum_{i_n=1}^{m} x_{i_1 i_2 \ldots i_n} = 1, \qquad \forall i_j = 1 \ldots m, \forall j = 2, \ldots, n-1 \tag{3–10}$$

$$\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_{n-1}=1}^{m} x_{i_1 i_2 \ldots i_n} = 1, \qquad \forall i_n = 1, \ldots, m \tag{3–11}$$

$$x_{i_1 i_2 \ldots i_n} \in \{0, 1\}, \qquad \forall i_1 = 1, \ldots, m, \ldots, i_n = 1, \ldots, m. \tag{3–12}$$

Surveys on the MAP and its applications can be found in Gilbert and Hofstra [60], Burkard and Çela [28], and more recently, Spieksma [117]. In addition, an excellent collection of articles on extensions of the classical assignment problem is given by Pardalos and Pitsoulis [97]. Another short introduction to assignment problems and their applications is presented by Çela [30]. Burkard and Çela [27] also give an annotated bibliography.

In this chapter, we consider the graph theoretical description of the MAP with decomposable costs. This implies that the cost of every assignment is a function of the pairwise assignment costs of the elements, or

$$c_{i_1 i_2 \ldots i_m} = f(c_{i_1 i_2}, c_{i_1, i_3}, \ldots, c_{i_{m-1} i_m}).$$

In Bandelt et al. [7], the most typically used decomposable cost functions are presented. One of them is the family of clique costs, which is often used for representing MAP assignment costs and is also followed by the authors herein. The clique cost of an assignment is considered to be the summation of all the edge costs that belong to the clique/assignment. Other families considered by the authors in [7] are minimum star, minimum tree, and minimum tour costs. One of the goals of this chapter is to show that

when considering a decomposable cost MAP, a significant amount of the computational time is spent calculating assignment/clique costs, based on the given edge costs as an input to the problem. Computational results will indeed verify that this is a serious bottleneck, which increases the runtime.

A number of methods has been proposed over the years to solve, either exactly or heuristically, the MAP. However, due to the inherent $\mathcal{NP}$–hardness of the problem, heuristic approaches have attracted the most practical interest. From those, we first note the high–quality solutions for large–scale instances provided by the Lagrangian relaxation for the MAP proposed by Poore and Robertson [108]. Branch–and–bound techniques have also been a viable option for solving MAPs, as shown in [84, 98, 100]. Furthermore, greedy randomized adaptive search procedures (GRASP [105] and GRASP with path–relinking [3]) and other randomized techniques [93] have proven to be efficient. Lastly, Pasiliao presents a series of local neighborhoods for the MAP in [99].

We consider the variant of the MAP with decomposable costs, where each assignment is represented by a clique. We propose two novel decomposition schemes that partition the problem into disjoint subproblems based on both the number of dimensions (m) and the number of elements per dimension (n). This partitioning will divide the feasible domain into smaller, more manageable sizes, which can be solved exactly or heuristically. Then, the solutions may be recombined in order to obtain upper and lower bounds to the original problem. Throughout the chapter, we refer to the set of dimensions as M, and the set of elements per dimension as N. Their respective sizes are considered to be $|M| = m$, and $|N| = n$, accordingly. Hence, whenever we refer to a full m–dimensional assignment problem with n elements per partition, we will denote it as either $AP(m, n)$, or $AP(M, N)$ interchangeably. Since the graph at hand is m–partite, we refer to each partition as $P_i, i = 1, \ldots, m$. Clearly, we have that the set of all elements is equal to $P_1 \cup P_2 \cup \cdots \cup P_m$.

The remainder of the chapter is outlined as follows: Section 3.2 introduces an element partitioning scheme; it shows that the solutions obtained are feasible and can serve as

upper bounds. Using this, we proceed to propose an exact (element augmentation) and a heuristic methodology (element partitioning) to solve the original MAP. In Section 3.3, we investigate a dimension partitioning scheme and prove that the solution obtained by the subproblems is a lower bound. We also present an exact methodology (dimension augmentation) and two heuristic approaches (2–AP, and dimension augmentation). Section 3.4 summarizes the results discussed in Sections 3.2 and 3.3; a hybrid method is also presented to take advantage of both decomposition schemes. Computational results are given in Section 3.5, which also concludes this study and offers insight on future work.

3.2 Element Decomposition

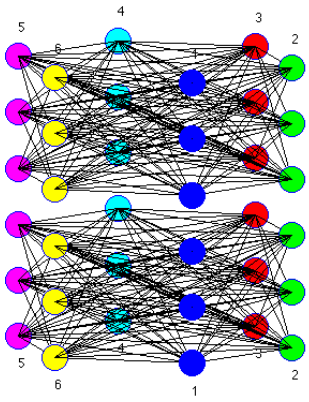
In this section, we discuss the first decomposition scheme, which partitions the MAP along its elements. We first present the idea of the decomposition and show that it provides us with a feasible solution (upper bound). Then, we proceed to present two algorithms, element augmentation and element partitioning, which provide us with an exact and heuristic solution, respectively, and analyze their worst–case time complexity.

3.2.1 Two disjoint subgraphs for element partitioning

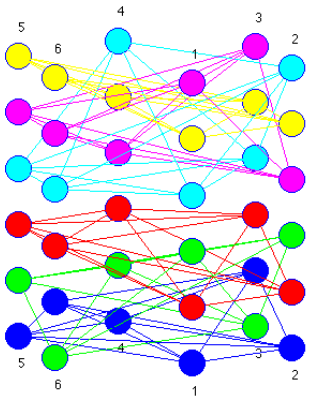
First of all, let us elaborate on the idea of decomposing the problem by presenting an example where the original m–partite graph is partitioned into two disjoint subgraphs. A simple way of decomposing the problem into two subproblems would be to halve the number of elements in each of them. This would result in two disjoint $K_{m\times n/2}$ subgraphs with corresponding optimization problems denoted by $AP(m, n/2)$. The decomposition is shown in Figure 3-1.

Observe that in the relaxed problem, all edges connecting elements from disjoint subsets are cut, and hence are not considered in any feasible solution of the problem. Also note that the solution obtained from the two subproblems is feasible to the original problem. We proceed to state this relationship between the solutions obtained from the subproblems and the original problem in Proposition 3.1.

Figure 3-1. Element Partitioning



A Two complete AP$(m, n/2)$



B Feasible solution of two AP$(m, n/2)$

Proposition 3.1. For any two nonempty element partitions N_1 and N_2 such that $N_1 \cap N_2 = \emptyset$ and $N_1 \cup N_2 = N$,

$$z^*\left[\mathrm{AP}(M,\ N_1)\right] \ +\ z^*\left[\mathrm{AP}(M,\ N_2)\right] \ \geq \quad z^*\left[\mathrm{AP}(M,\ N_1 \cup N_2)\right]$$

where M is the set of dimensions, N is the set of elements, and z^* is the optimal value of a given assignment problem.

Proof. Consider $G'(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}' = N_1 \cup N_2 = N$, and $\mathcal{E}' = (N_1 \times N_1) \cup (N_2 \times N_2)$. Observe that we have $\mathcal{E} \supseteq \mathcal{E}'$, hence for an optimal solution over the original edge set, we obtain

$$z^*(AP(M, N)) \leq z^*(G'(\mathcal{V}', \mathcal{E}')).$$

On top of that, it is easy to see that

$$z^*(G'(\mathcal{V}', \mathcal{E}')) = z^*(AP(M, N_1)) + z^*(AP(M, N_2)).$$

Combining, we have the result.

□

Proposition 3.1 remains true regardless of the number of the partitioned subproblems, or the number of elements per partition. The only requirement for the proposition to hold

is that, for k subproblems, we have,

$$N_1 \cap N_2 \cap \cdots \cap N_k = \emptyset.$$

By decomposing the problem, we see that each subproblem can be solved faster. Formally, the original problem $AP(m, n)$, has a worst–case time complexity of $\mathcal{O}((n!)^{m-1})$. On the other hand, the two subproblems are still hard to solve, but with a worst–case time complexity of $\mathcal{O}((n/2!)^{m-1})$ each. Hence, we can come up faster with a feasible solution (cf. Proposition 1), and obtain an upper bound. This upper bound will be used in our experiments as a bound for an implicit enumeration algorithm.

3.2.2 Element augmentation

Further expanding on the idea presented in the previous subsection, we propose an iterative algorithm. In each iteration, we consider an extra element per dimension, until the original problem is solved. Let $S = \{P_1, P_2, \ldots, P_m\}$, where P_i refers to the set of elements of dimension i (partition i). Now, we can present Element Augmentation in Algorithm 1.

Algorithm 1 Element Augmentation

```
n ← 2
for i = 1 → m do
    N ← {u, v} ∈ P_i
    S ← S/{u, v}
end for
while S ≠ ∅ do
    Solve AP(m, n) over N, using ẑ(AP(m, n − 1)) as UB

    for i = 1 → m do
        N ← {u} ∈ P_i
        S ← S/{u}
    end for
    n ← n + 1
end while
return ẑ
```

The algorithm begins with a very simple to solve $AP(m, 2)$ with only two elements per dimension. The optimal solution of the problem would present us with a set of 2

disjoint cliques. We then proceed to add another element (at random) from the set of elements per partition that have not been considered. The new problem can again be solved to optimality, using the set of cliques obtained in the previous iteration as a starting solution.

Observe that, in theory, the worst–case time complexity of the approach presented in Algorithm 1 (Element Augmentation) is worse than the one derived for the original problem. This complexity is shown in Equation 3–13.

$$\begin{aligned} \mathrm{T}(m,n) &= 2^{(m-1)} + 6^{(m-1)} + \cdots \\ &\quad + [(n-1)!]^{(m-1)} + [n!]^{(m-1)} \end{aligned} \tag{3–13}$$

The increased complexity can be attributed to the fact that instead of solving one full m–dimensional assignment problem, we solve a series of subproblems of increasing size, $AP(m,2)$, $AP(m,3)$, $\ldots$, $AP(m,n)$. As it will be shown in the numerical experiments, though, the performance of the algorithm is significantly better, due in part to the tighter upper bounds and the high quality initial solution at each subproblem.

3.2.3 Element partitioning

Another approach would be to immediately create a series of k partitions of the original problem, that would then be combined to provide us with an upper bound. Such a randomized approach would be faster to provide us with a feasible solution. However, the quality of the solution would be dependent on the initial selection of the partitions. The methodology (Element Partitioning) is described in Algorithm 2.

At first, we randomly select k similarly sized partitions. Note that the function denoted as $rand(n,N)$ is essentially selecting n elements from each set of dimensions. In case n/k is not integer, the result is truncated and the last subproblem is allowed to have more than n/k elements per partition. Last, observe that it is important for the subproblems to be of similar size, in order to ensure that no subproblem is much harder to solve.

Algorithm 2 Element Partitioning

```
k ← k_0
for i = 1 → k do
    P_i ← rand(n/k, N)
    N ← N/P_i
end for
for i = 1 → k do
    ẑ_i ← AP(M, n/k) over P_i
    ẑ ← [ẑ  ẑ_i]
end for
return ẑ
```

Assuming that all subproblems have the same size (n/k), the overall worst–case time complexity of the approach in Algorithm 2 is given in Equation 3–14.

$$\mathrm{T}(m,n) \quad = \quad k \cdot ((n/k)!)^{(m-1)}. \tag{3–14}$$

We will show in the computational results, that this approach is significantly faster, however the quality of results is worse, mainly because of the random selection of partitions.

3.2.4 Divide and conquer

Finally, the element partition approach can be implemented as a recursive divide and conquer algorithm to either find bounds to the m-dimensional assignment problem or as a method for solving the problem exactly. Although the divide and conquer approach may actually solve the worst-case assignment problem in longer time, this approach provides a tighter upper bound for the partial solutions.

Recursive Relation. The divide and conquer approach is described by the recursive relation given below:

$$T(m,n) \quad = \quad 2 \times T(m,n/2) + (n!)^{m-1} \tag{3–15}$$

The MAP is iteratively broken into two halves until all the subproblems are of type $\mathrm{AP}(m,2)$. From there, we begin to construct the complete solution by solving a set of $\mathrm{AP}(m,4)$ problems by using the solutions of two $\mathrm{AP}(m,2)$ problems with the same

elements as upper bounds. We continue combining solutions until we can solve the original AP(m, n) by using the solutions of two AP$(m, n/2)$ as upper bounds.

3.2.5 Preprocessing

The performance of each of the approaches is dependent on how well we can break apart the elements in the dimensions. We can simply and quickly partition the dimensions by arbitrarily or randomly choosing which dimensions go in each partition. However, we can significantly improve the performance of the algorithms by partitioning wisely. Ideally, we would like to partition the elements in such a way that the edges of an optimal solution are not cut. In other words, we want all the elements of a clique in an optimal solution to be in the same subproblem.

An approach would be to quickly find a good feasible solution through a fast heuristic and then, use the heuristic solution as a guide for partitioning the elements. No edges that are in the feasible solution are to be cut. The element partition approach should result in a faster convergence rate as the initial heuristic solution improves. In the case where the heuristic happens to find an optimal solution, then all of the element partition procedures would converge extremely fast since no reassignments are necessary. The procedures would then provide a polynomial time algorithm for checking the optimality of the solution.

For the Element Augmentation approach, we can preprocess the data set such that the elements with the lower clique costs are placed at the top positions. The rearrangement of the elements in each dimension should satisfy the condition that the clique defined by the top element in each dimension is lower than the clique defined by the second in each dimension. The clique from the second elements is of lower cost than from the third elements, and so on. Preprocessing the data set to satisfying this condition will decrease the likelihood of a that a clique will be removed from the current solution when new elements are added to the problem. This approach is actually the purely greedy approach in which the lowest cost feasible cliques are placed in the solution. The goal

is to maximize the intersection between the clique set solutions of AP$(m, n-1)$ and of AP(m, n) to tighten the bounds and increase the algorithm's convergence rate.

3.2.6 Discussion

This section presented different approaches to partitioning the elements of an m-partite graph to solve smaller problems, which lead to upper bounds. Section 3.2.1 describes the methodology for partitioning the original AP(m, n) into two smaller instances of AP$(m, n/2)$. Section 3.2.2 shows how we can start from the basic AP$(m, 2)$ and sequentially augmenting the number of elements until we have the complete solution for AP(m, n). Section 3.2.4 illustrates how we can combine characteristics of the Element Partition and Augmentation approaches to implement a divide and conquer procedure for solving AP(m, n). The divide and conquer approach continually partitions the problem until we have a set of subproblems of size AP$(m, 2)$. We then sequentially augment the number of elements by combining two similarly sized subproblems at a time.

3.3 Dimension Decomposition

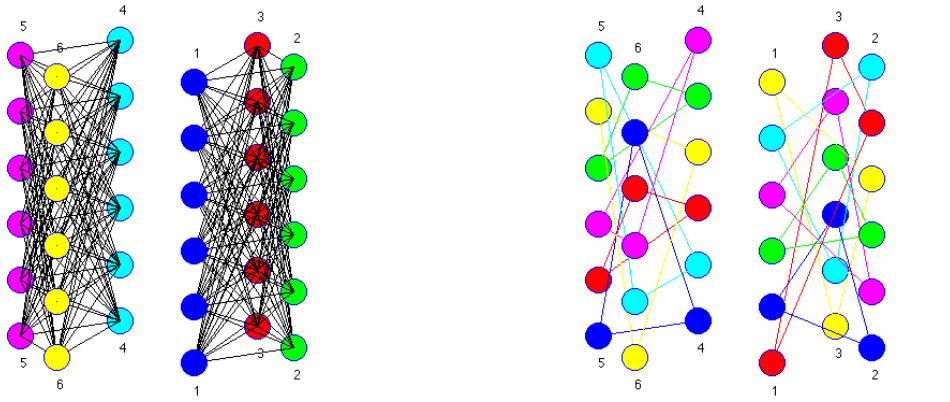
In this section, we partition the dimension set of the MAP, in order to obtain subproblems of smaller dimensionality. This will decrease the computational time required to solve each subproblem. However, any problem obtained by a valid dimension decomposition is infeasible to the original problem. Hence, it can serve as a lower bound. We investigate an exact method (dimension augmentation), which is similar to the element augmentation presented before, in its solving iteratively subproblems of larger dimensionality until the original MAP is solved. Then, we propose two heuristic methods. The first one considers two dimensions at a time, solving a series of (polynomially solvable) $AP(2, n)$, while the other partitions the set into disjoint subproblems and then matches the cliques obtained to obtain a feasible solution.

3.3.1 Two disjoint subgraphs for dimension partitioning

Let us begin, similarly to before, by considering two disjoint subproblems. Each of them can be represented as $AP(m/2, n)$. Observe that in the two subproblems, the

number of feasible solutions (cliques) is significantly reduced to $2 \times (n!)^{m/2-1}$ from the original $(n!)^{m-1}$. Such a decomposition is shown in Figure 3-2.

Figure 3-2. Dimension Partitioning



A Two complete AP$(m/2, n)$ B Feasible solution of two AP$(m/2, n)$

In the example of Figure 3-2, an optimal solution to the two subproblems yields a total of twelve disjoint cliques of smaller size. In the original MAP, though, we require a feasible solution to consist of six disjoint cliques, spanning through all dimensions. Hence, any solution obtained from the two subproblems, is infeasible to the original problem. This is established in Proposition 3.2.

Proposition 3.2. For any two nonempty dimension partitions M_1 and M_2 such that $M_1 \cap M_2 = \emptyset$,

$$z^*\left[\mathrm{AP}(M_1\ , N)\right]\ +\ z^*\left[\mathrm{AP}(M_2,\ N)\right]\ \leq\ z^*\left[\mathrm{AP}(M_1 \cup M_2,\ N)\right]$$

where M is the set of dimensions, N is the set of elements, and z^* is the optimal value of a given assignment problem.

Proof. The optimal solution for a partition $M_1, z^*\left[AP(M_1\ , N)\right]$ contains $|N|$ disjoint cliques, each of a size of $|M_1|$. Similarly, $z^*\left[\mathrm{AP}(M_2\ , N)\right]$ would contain $|N|$ disjoint cliques of size $|M_2|$. Note that in order to transform this solution into a feasible solution for the original problem, we would need to reconcile the cliques and create $|N|$ disjoint cliques

of size $|M_1 \cup M_2|$. This implies that at least some edges connecting the $2|N|$ disjoint cliques need to be added. Since the costs between any two nodes are nonnegative, that also implies that

$$z^* \left[\mathrm{AP}(M_1\ , N)\right] + z^* \left[\mathrm{AP}(M_2\ , N)\right] \leq \hat{z} \left[\mathrm{AP}(M_1 \cup M_2\ , N)\right],$$

where $\hat{z} \left[\mathrm{AP}(M_2\ , N)\right]$ is a feasible solution to the $AP(M_1 \cup M_2, N)$. Last, since this is a feasible solution, we can easily obtain that

$$\hat{z} \left[\mathrm{AP}(M_1 \cup M_2\ , N)\right] \leq z^* \left[\mathrm{AP}(M_1 \cup M_2,\ \ N)\right],$$

and the result is at hand.

□

3.3.2 Dimension augmentation

The Dimension Augmentation algorithm begins, as in the Element Augmentation, by selecting two dimensions of the problem. The subproblem is a 2-dimensional assignment problem which is easily solved in polynomial time. In the next iteration, we proceed to add a new dimension to the problem. The new subproblem is still a 2–dimensional assignment problem, which matches the cliques obtained in the previous iteration, with the elements of the incoming dimension, by considering the cost of adding the elements to the existing cliques. We iterate on that, until all dimensions have been successfully added to the problem. Observe that at every iteration only an $AP(2, n)$ is solved. Because of that, a feasible solution can be readily found, without the need to fully enumerate all cliques/assignments in the original problem. This approach is presented in Algorithm 3.

Contrary to the element augmentation approach, Algorithm 3 has a worst–case time complexity (shown in Equation 3–16) that is much smaller than the one of the original $AP(m, n)$.

$$\mathrm{T}(m, n) \ \ = (m-1) \times \mathcal{O}(n^3) \tag{3–16}$$

Algorithm 3 Dimension Augmentation (2-AP solver)

$i \leftarrow rand(1, \ldots, m)$
$j \leftarrow rand(1, \ldots, m)$
$N \leftarrow P_i \cup P_j$
$S \leftarrow S/\{P_i, P_j\}$
while $S \neq \emptyset$ do
    Solve $AP(2, n)$
    $N \leftarrow \hat{z}(AP(2, n))$
    $k \leftarrow rand(i : P_i \in S)$
    $N \leftarrow N \cup P_k$
    $S \leftarrow S/P_k$
end while
return $\hat{z}$

On the other hand, instead of continuously solving 2–dimensional assignment problems between the previous set of cliques and the incoming dimension's elements, we could proceed as follows. After obtaining the solution of the relaxed subproblem $AP(2, n)$, we can go ahead and use the lower bound obtained to solve exactly the $AP(3, n)$ with the added new dimension. This would result, as it can be clearly seen, in a much worse time complexity (see Equation 3–17). This is to be expected, since we solve exactly each of the $AP(2, n), AP(3, n), \ldots, AP(m, n)$. The pseudocode is presented in Algorithm 4.

$$\begin{aligned} \mathrm{T}(m, n) &= 2^{(m-1)} + 6^{(m-1)} + \cdots \\ &\quad + [(n-1)!]^{(m-1)} + [n!]^{(m-1)} \end{aligned} \tag{3–17}$$

The performance of each of the dimension partition approaches is dependent on how well we can break apart the problem into dimension subsets. We can arbitrarily and randomly select how the dimensions will be partitioned. However, we can improve the performance of the algorithms by partitioning wisely.

Similarly to the Element Partition preprocessing, we can perform a fast heuristic to find a good solution before partitioning the dimensions. We would then look to partition the dimensions such that the cut edges are of highest total costs. A "brute" force approach is to simply sum all the edge costs from one dimension to another. If the total

Algorithm 4 Dimension Augmentation (exact)

$m_0 \leftarrow 2$
$i \leftarrow rand(M)$
$M \leftarrow M/\{i\}$
$j \leftarrow rand(M)$
$M \leftarrow M/\{j\}$
$N \leftarrow P_i \cup P_j$
$S \leftarrow S/\{P_i, P_j\}$
while $S \neq \emptyset$ do
    Solve $AP(m_0, N)$, using $\hat{z}(AP(m_0 - 1, N))$ as LB
    $i \leftarrow rand(M)$
    $M \leftarrow M/\{i\}$
    $N \leftarrow N \cup P_i$
    $S \leftarrow S/\{P_i\}$
    $m_0 \leftarrow m_0 + 1$
end while
return $\hat{z}$

is high, then we place the dimensions in different partitions. In other words, we place the dimensions with a low total edge costs between them in the same partition.

In a multi-target multi-sensor tracking application, we would like to group together the dimensions or sensors that are physically close to each other. This is because if one of the sensors picked up a target, there is a good chance that other sensors close by would have also picked up the same target. For the multi-target multi-scan tracking application, we want to group the dimensions or scans that are temporally close to each other. Again, this is so that a target that is picked up in one scan has a high probability of also being picked up by the scans immediately before and after the current scan. From a purely mathematical perspective, group the dimensions that are most compatible with each other in a some sense. The goal is to partition the dimensions in such a way so that the optimal clique solutions from two different partitions stay intact when the partial problems are combined. One approach is to use the 2AP Relaxation described before. The pairwise matching provides a measure of "affinity" between two dimensions. A low overall solution cost indicates a strong "affinity" between two dimensions. From there, we essentially perform a clustering of the the dimensions into partitions. It may well be that a

non-balanced partitioning, where not all partitions have the same number of dimensions, is more appropriate. This approach would improve the chances that the cliques found in the partitions will not be broken up when the partial solutions are combined.

Ideally, we would like to partition the dimensions in such a way that the optimal clique solutions of the subproblems are subsets of the overall optimal solution. In other words, we want all the elements of a clique in an optimal subproblem solution to be in the optimal complete solution. An approach would be to quickly find a good feasible solution through a fast heuristic and to use the heuristic solution as a guide for partitioning the elements.

The dimension partition approach should result in a faster convergence rate as the initial heuristic solution improves. In the case where the subproblem cliques are all in the same subset the heuristic happens to find an optimal solution and then all of the element partition procedures would converge extremely fast since no reassignments are necessary. The procedures would then provide a polynomial time algorithm for checking the optimality of the solution.

3.3.3 Dimension partitioning

We now present a dimension partitioning scheme, based on a divide–and–conquer algorithm. Observe that halving the problem in two $AP(m/2, n)$ subproblems, significantly decreases the runtime. However, as shown in Proposition 2, the obtained solution is infeasible. In this subsection we discuss a way to reconcile the cliques of each subproblem, hence obtaining a feasible solution to the initial $AP(m, n)$.

The algorithm divides the original problem into k disjoint subproblems. Then the solutions obtained are reconciled, using the Reconcile Cliques subroutine shown in Algorithm 6. The pseudocode of this approach is presented in Algorithm 5.

In the case of just two disjoint subproblem, each represented by $AP(m/2, n)$, the overall worst–case time complexity is derived in Equation 3–18.

Algorithm 5 Dimension Partitioning

$k \leftarrow k_0$
for $i = 1 \rightarrow k$ do
  $M_i \leftarrow rand(m/k, M)$
  $M \leftarrow M/M_1$
end for
for $i = 1 \rightarrow k$ do
  $\hat{z}_i \leftarrow AP(M_i, N)$
end for
$\hat{z} \leftarrow Reconcile(z_1, z_2, \ldots, z_k)$
return $\hat{z}$

$$T(m,n) \quad = \quad 2 \times T(m/2, n) + \mathcal{O}(n^3) \tag{3–18}$$

In Algorithm 5, we refer to a subroutine called Reconcile, which is presented in Algorithm 6. After solving two subproblems, $AP(M_1, N)$ and $AP(M_2, N)$, their solution sets can be reconciled in order to provide us with one set of larger cliques. That solution will be feasible to the problem $AP(M_1 \cup M_2, N)$.

Algorithm 6 Reconcile Cliques

$M_1 \leftarrow Q_1 \subseteq M$
$M_2 \leftarrow Q_2 \subseteq M : Q_1 \neq Q_2$
$D_1 \leftarrow AP(M_1, n)$
$D_2 \leftarrow AP(M_2, n)$
$\hat{z} \leftarrow AP(2, D_1 \cup D_2)$
return $\hat{z}$

Observe that the time complexity of computing all possible assignments between the cliques in the hyperset D_1 and those in D_2 is $\mathcal{O}(m^2 n^2)$.

The worst–case time complexity of the algorithm for partitioning the dimensions into two subsets, solving each subproblem separately, and reconciling the cliques from the subproblems is given by the following expression:

$$\mathrm{T}(m,n) \quad = \quad 2 \times (n!)^{(m/2-1)} + (m/2)^2 \times n^2 + \mathcal{O}(n^3)$$

3.4 Analysis and a Hybrid Method

3.4.1 Analysis

In the previous sections of the chapter, we introduced two novel decomposition schemes for the classical MAP. For each of them, both exact and heuristic approaches were discussed. In summary, we proposed the Element Augmentation and Dimension Augmentation (exact) algorithms to optimally solve the problem, and the Dimension Augmentation (2-AP), Element and Dimension Partitioning algorithms to obtain a heuristic solution. This distinction is reflected in Table 3-1, where the worst-case time complexities of the original MAP and our approaches are concerned.

Observe that both the Element and Dimension (exact) Augmentation algorithms have a worse overall time complexity. However, as it will be shown in the computational results, the methods still outperform in general the classical MAP formulation. This is, in part, due to bounds obtained from the previous iterations (the smaller subproblems), which lead to faster convergence. On the other hand, the rest of the approaches (Element/Dimension Partitioning, Dimension Augmentation (2-AP)) have a better worst–case time complexity, which is expected, since they provide a heuristic solution. The quality of these solutions, along with the runtimes will be discussed in the following section.

Table 3-2 presents some useful properties, that have to do with the structure of the graph considered for each of the partitioning schemes. It is interesting to note the decrease in the number of cliques that need to be considered, when the original problem is decomposed into disjoint subgraphs.

Based on these remarks, we can now proceed to introduce a hybrid algorithm that employs both partitioning schemes, in an attempt to faster converge to a high quality solution to the original MAP.

Table 3-1. Worst-case time complexity of the approaches presented.

- Explicit Enumeration

$$\mathrm{T}(m,n) \;=\; (n!)^{(m-1)}$$

- Element Augmentation

$$\mathrm{T}(m,n) \;=\; 2^{(m-1)} + 6^{(m-1)} \;+\; \cdots \;+\; [(n-1)!]^{(m-1)} \;+\; [n!]^{(m-1)}$$

- Element Partitioning (over k partitions)

$$\mathrm{T}(m,n) \;=\; k \cdot (n/k!)^{(m-1)}$$

- Dimension Augmentation (2–AP solver)

$$\mathrm{T}(m,n) \;=\; (m-1) \times \mathcal{O}(n^3)$$

- Dimension Augmentation (exact solver)

$$\mathrm{T}(m,n) \;=\; \mathcal{O}(n^3) \;+\; [\mathcal{O}(n^3) + (n!)^2] \;+\; \cdots \;+\; \left[\mathcal{O}(n^3) + (n!)^{(m-2)}\right] + \left[\mathcal{O}(n^3) + (n!)^{(m-1)}\right]$$

- Dimension Partitioning (2 disjoint subgraphs)

$$\mathrm{T}(m,n) \;=\; (m-1) \times \mathcal{O}(n^3)]$$

3.4.2 Hybrid method

The hybrid method employs both dimension and element decomposition schemes to faster solve large–scale (i.e., high dimensionality and/or bigger cardinality of partitions). The algorithm is presented in Algorithm 7.

The key advantage of the method is that it provides us with a systemic way of decomposing large–scale MAPs, by employing the lower and upper bounds obtained at every iteration. Based on these bounds, we obtain an optimality gap (and a feasible solution) that is significantly smaller (of higher quality) compared to the element and

Table 3-2. Properties of the $AP(m,n)$ and its decomposition schemes.

•	Edges	$\frac{1}{2}\,m(m-1)\times n^2$	One AP(m,n)
		$2\times\frac{1}{2}\,m(m-1)\times\left(\frac{n}{2}\right)^2$	Two AP$(m,n/2)$
		$2\times\frac{1}{2}\,\left(\frac{m}{2}\right)\left(\frac{m}{2}-1\right)\times n^2$	Two AP$(m/2,n)$
•	Cliques	n^m	One AP(m,n)
		$2\times\left(\frac{n}{2}\right)^m$	Two AP$(m,n/2)$
		$2\times n^{\left(\frac{m}{2}\right)}$	Two AP$(m/2,n)$
•	Feasible Solutions	$[\,n!\,]^{(m-1)}$	One AP(m,n)
		$\left[\,\left(\frac{n}{2}\right)!\,\right]^{[2\times(m-1)]}$	Two AP$(m,n/2)$
		$[\,n!\,]\times[\,n!\,]^{\left(\frac{m}{2}-1\right)}$	Two AP$(m/2,n)$

Algorithm 7 Hybrid Algorithm

ElementPartition$(AP(m,n))$ into k partitions
for $i=1\rightarrow k$ do
    DimensionPartition$(AP(m,n_i))$
end for
$\hat{z}=$ *ReconcileCliques*$(\hat{z}_1(AP(2,n_1)),\hat{z}_2(AP(2,n_2)),\ldots,\hat{z}_k(AP(2,n_k)))$
return $\hat{z}$

dimension partitioning, when employed by themselves. On top of that, as it will be shown in the computational results, this method can solve larger problems than the original IP formulation of the MAP.

3.5 Computational Results

In this section, we will depict the success of our decomposition schemes by solving a series of 4-dimensional, 6-dimensional, and 8-dimensional assignment problems. All

experiments were performed on an Intel Core 2 Duo at 2.4 GHz with 4 GB of RAM. All codes were written in C++ and the results obtained were contrasted against the results of Gurobi 5.0.1 and CPLEX 12.4. The implemented algorithms, as well as the commercial solver, were allowed to utilize both cores, thus speeding up the process. A GPU implementation was also created, however it did not prove to be computationally efficient since the solvers could not use the extra resources to find optimal solutions to the decomposed/original problems.

The goal of our experiments was to show how well our decomposition algorithms (both per dimension and per element) perform. Another objective was to show that calculating clique (assignment) costs is a serious bottleneck of the problem. Especially in larger instances, a significant fraction of the operations time was devoted to computing these costs. In our approaches, we discard a large number of cliques per subproblem and, hence, the computations are much faster. This will be better shown in Table 3-3.

Before we proceed to the computational results though, let us describe the way that the testbed was set. For each of the experiments, 50 random instances of each size were created. These 50 instances were then solved by a commercial solver (Gurobi 5.0.1 and CPLEX 12.4) and the fastest (or the one with the smallest optimality gap, if not solved optimally) solution time was reported. All instances were randomly generated, following a uniform distribution, based on the instance generator presented in [62]. In the paper, the authors show that these instances are hard to solve, and how they can serve as benchmark instances.

First, all instances were partitioned into smaller subproblems per element. In this implementation, $n/5 - n/10$ elements per partition were considered at random. Clearly, employing a smarter way to define the partitions would yield better results, however it was our intention to test how well large-scale MAPs can be solved even when using a random partitioning scheme. After obtaining the smaller subproblems, we tackle each instance using the algorithms of Element Augmentation and Element Partitioning.

Then, we partitioned the original problems on the set of dimensions. For the Dimension Augmentation approach, both the 2-AP solver, and the exact methodology were applied and compared. In addition to that, we solved the problems, utilizing the Dimension Partitioning method. For the latter, all possible $\frac{\binom{m}{2}}{2}$ dimension pairings were considered. Last, we contrast all results to the ones obtained by the hybrid method, and the exact solver (CPLEX and Gurobi).

The instances are denoted by mDn, where m is the number of dimensions of the MAP, and n the number of elements per partition.

Table 3-3. Time spent (in seconds) computing clique costs.

Problem Instance	Element Partitioning	Dimension Partitioning	Full Clique Computation
4D5	0.003	0.02	0.12
4D10	0.004	0.03	0.52
4D15	0.004	0.04	2.55
4D20	0.004	0.45	7.82
4D25	0.005	0.88	50.53
4D30	0.005	1.07	493.85
4D35	0.01	1.95	711.21
4D40	0.01	5.63	988.43
4D45	0.01	11.33	1076.04
4D50	0.01	14.10	1386.92
6D5	0.004	0.10	0.24
6D10	0.005	0.38	0.98
6D15	0.006	0.79	3.95
6D20	0.01	1.13	11.07
6D25	0.01	4.07	80.12
6D30	0.02	7.81	713.50
6D35	0.02	13.19	1109.02
6D40	0.02	19.75	1533.24
8D5	0.006	0.14	0.55
8D10	0.01	0.67	3.76
8D20	0.02	7.94	981.47

Since the input of the problems was the set of all edge costs among the nodes belonging to different partitions of the m–partite graph, some time needs to be devoted to build the assignment costs, before proceeding to solving the problem. Table 3-3 presents

our findings on the computational time required to compute all clique/assignment costs, when they are provided in decomposed form (i.e., per edge), in the original problem, and in the subproblems after partitioning the graph at hand in both the elements and dimensions. It is easy to see that in the full MAP, computing all possible cliques of size m (feasible assignments) in the graph is computationally expensive, especially as the number of elements per partition increases. On the other hand, decomposing the problem in both the elements and the dimensions discards a significant number of edges, hence decreasing the overall computational time required to calculate all assignment costs.

In Table 3-4, we present our findings for the efficiency, and the quality of solutions obtained for the two partitioning schemes, and each of the methods devised for them. Runtimes are in seconds, while the optimality gaps are given in parentheses (when present). First, we partitioned the original problem in its elements, and solved the problem exactly (Element Augmentation) and heuristically (Element Partitioning). Then, partitioning the problem in its dimensions gives us the opportunity to test the three methods: 2-AP Dimension Augmentation (heuristic), Dimension Augmentation (exact), and Dimension Partitioning (heuristic). Last, we present the computational runtimes obtained by the hybrid method of Section 3.4, and the commercial solver (CPLEX and Gurobi).

Table 3-4. Runtimes and optimality gaps for the exact and heuristic approaches investigated.

Problem Instance	Element Augmentation	Element Partitioning	Dimension Augmentation	Dimension Augmentation (2–AP)	Dimension Partitioning	Hybrid Method	Commercial Solver
4D5	0.21	0.00 (19.02%)	0.07	0.01 (18.72%)	0.01 (7.06%)	0.01 (5.12%)	0.13
4D10	0.31	0.01 (21.05%)	0.22	0.03 (18.36%)	0.04 (10.11%)	0.04 (7.81%)	0.27
4D15	2.97	0.02 (19.95%)	2.25	0.04 (19.04%)	0.06 (9.52%)	0.09 (9.99%)	3.35
4D20	60.11	0.50 (19.26%)	51.06	0.10 (19.50%)	0.09 (14.97%)	0.32 (10.30%)	77.02
4D25	234.43	1.01 (29.32%)	201.33	0.16 (19.51%)	0.16 (16.24%)	0.98 (11.88%)	271.65
4D30	990.72	1.71 (30.18%)	1,103.28	0.26 (23.25%)	0.25 (17.57%)	1.12 (12.37%)	1,106.84
4D35	1,284.35	2.15 (36.35%)	1,743.90	0.54 (24.48%)	0.55 (29.87%)	2.90 (15.10%)	1,663.70
4D40	1,597.44	3.96 (38.14%)	2,411.20	2.81 (29.11%)	2.76 (18.20%)	4.11 (17.63%)	2,306.19
4D45	7,668.47	7.10 (27.71%)	14,512.11	3.17 (31.94%)	3.05 (27.71%)	5.67 (18.12%)	12,649.90
4D50	15,331.95 (0.03%)	13.23 (34.12%)	21,600.00 (8.60%)	6.09 (33.05%)	5.29 (23.97%)	9.03 (20.90%)	18,835.00 (1.02%)
6D5	0.91	0.01 (27.23%)	0.46	0.05 (19.98%)	0.02 (4.54%)	0.01 (2.24%)	0.59
6D10	6.33	0.03 (11.90%)	3.24	0.41 (21.35%)	0.34 (12.09%)	0.09 (7.83%)	4.09
6D15	407.11	0.60 (17.18%)	366.25	1.17 (21.50%)	0.87 (17.45%)	0.78 (12.14%)	405.70
6D20	1,821.90	1.89 (21.44%)	1,577.68	1.36 (22.47%)	1.08 (21.78%)	0.50 (15.04%)	1,900.12
6D25	2,601.87	2.70 (21.94%)	2,701.20	4.02 (23.03%)	1.69 (22.91%)	1.11 (16.30%)	2,713.11
6D30	15,976.10	8.56 (27.08%)	21,600.00 (1.25%)	7.74 (24.77%)	3.17 (24.43%)	4.89 (19.11%)	18,912.80 (0.13%)
6D35	19,304.42 (0.08%)	10.13 (27.34%)	21,600.00 (2.07%)	11.08 (31.20%)	4.96 (27.78%)	6.01 (19.97%)	21,600.00 (0.91%)
6D40	21,600.00 (0.83%)	14.97 (33.89%)	21,600.00 (6.16%)	16.11 (36.81%)	6.89 (31.05%)	8.30 (21.00%)	21,600.00 (1.37%)
8D5	2.13	0.01 (23.14%)	1.09	0.08 (20.09%)	0.04 (9.82%)	0.03 (5.08%)	1.67
8D10	1,320.9	1.04 (18.36%)	1,012.38	2.15 (16.40%)	1.39 (23.91%)	2.07 (19.44%)	1,324.5
8D20	21,600 (1.39%)	17.12 (34.91%)	21,600 (1.89%)	22.85 (32.74%)	15.38 (31.09%)	17.70 (28.81%)	21,600 (2.03%)

Table 3-5. Average optimality gaps reported if each of the methods was allowed a runtime of at most 18 seconds.

Problem Instance	Element Augmentation	Element Partitioning	Dimension Augmentation	Dimension Augmentation (2–AP)	Dimension Partitioning	Hybrid Method	Commercial Solver
4D5	0%	19.02%	0%	18.72%	7.06%	5.12%	0%
4D10	0%	21.05%	0%	18.36%	10.11%	7.81%	0%
4D15	0%	19.95%	0%	19.04%	9.52%	9.99%	0%
4D20	3.34%	19.26%	2.19%	19.50%	14.97%	10.30%	4.25%
4D25	8.89%	29.32%	8.02%	19.51%	16.24%	11.88%	9.03%
4D30	18.33%	30.18%	21.07%	23.25%	17.57%	12.37%	20.27%
4D35	22.47%	36.35%	29.76%	24.48%	29.87%	15.10%	32.98%
4D40	36.59%	38.14%	43.17%	29.11%	18.20%	17.63%	41.70%
4D45	40.55%	37.71%	59.61%	31.94%	27.71%	18.12%	62.20%
4D50	49.21%	34.12%	83.11%	33.05%	23.97%	20.90%	79.30%
6D5	0%	27.23%	0%	19.98%	4.54%	2.24%	0%
6D10	0%	11.90%	0%	21.35%	12.09%	7.83%	0%
6D15	5.61%	17.18%	5.44%	21.50%	17.45%	12.14%	7.33%
6D20	19.40%	21.44%	15.11%	22.47%	21.78%	15.04%	19.56%
6D25	27.12%	21.94%	27.91%	23.03%	22.91%	16.30%	31.08%
6D30	48.75%	27.08%	62.09%	24.77%	24.43%	19.11%	59.87%
6D35	55.23%	27.34%	76.44%	31.20%	27.78%	19.97%	74.21% s
6D40	80.91%	33.89%	94.11%	36.81%	31.05%	21.00%	79.52%
8D5	0%	23.14%	0%	20.09%	9.82%	5.08%	0%
8D10	9.82%	18.36%	9.97%	16.40%	23.91%	19.44%	11.06%
8D20	41.56%	34.91%	46.12%	32.74%	31.09%	28.81%	39.15%

In the biggest of instances, the commercial solvers were unable to solve all of them within the limit of 6 hours (21600 seconds). For these instances, the best solution obtained by either Gurobi or CPLEX was reported and, hence, there is also an optimality gap associated with them. The average optimality gap for each method is presented (when it exists) in parentheses.

From Table 3-4, we note the following. The methodologies presented for tackling the problem exactly outperform the commercial solver in most instances. It is interesting to observe that Dimension Augmentation is faster than Element Augmentation in the instances with a smaller number of elements per partition. However, as the number of elements increases, Element Augmentation is more efficient, as expected from the complexity analysis. On the other hand, the heuristic approaches devised, provide us with high quality solutions, especially when weighing in their extremely fast runtimes. From the heuristic approaches, the Hybrid Method has consistently outperformed the rest as far as the average optimality gaps reported are concerned.

Another interesting factor that we weighed in our experiments has to do with the quality of the solutions obtained by the commercial solver if allowed to run for exactly the same time as our approaches. These results are given in Table 3-5. Observe that in most instances, the optimality gap of the commercial solver is larger than the ones obtained by the rest of the methods. Hence, when a high quality feasible solution is required fast, the heuristic methods presented here pose a better alternative.

In summary, we see that, especially as the size of the instances increases, the commercial solvers become less and less practical, as expected. On the other hand, there is only a small computational overhead in the proposed methodologies. Of course, the solutions obtained are suboptimal, however the efficacy of our approaches can be seen in the computational times achieved. In order to further show how the problem scales with size, and how the quality of the solutions obtained by commercial solvers significantly drops, we present three exemplar large-scale instances (namely 10D100, 20D50, and

Table 3-6. Very large-scale instances.

Problem Instance	Element Partitioning	Dimension Augmentation (2–AP)	Dimension Partitioning	Hybrid Method
10D100	121.93 (39.17%)	147.03 (38.57%)	201.33 (27.13%)	205.76 (25.92%)
20D50	204.77 (41.05%)	254.31 (38.36%)	390.42 (30.11%)	401.93 (26.81%)
30D30	267.31 (39.95%)	282.91 (39.04%)	448.83 (34.52%)	572.88 (29.32%)

30D30), where the commercial solver was unable to provide us with a solution, due to memory restrictions. All four of our heuristic approaches provided us with a feasible solution within – at most – minutes. It is interesting to note that the commercial solvers could not tackle these problems, as they ran out of memory. In parentheses, we present the average optimality gap associated with each approach. These results are given in Table 3-6.

To sum up, two decomposition schemes for the MAP are presented. As shown in Section 3.5, the numerical experiments prove that our algorithms perform well, especially in large-scale instances. In addition to that, both decomposition schemes admit parallel implementations, which implies that with the appropriate computational power the results can be further enhanced. Since the MAP is, in general, a non–separable optimization problem, techniques that can decompose the problem into smaller ones are inherently of high quality, and can reach suboptimal solutions in an efficient and timely manner. Overall, our techniques have proved to be of interest in solving multi-target multi-sensor and/or data association problems with near optimal solutions.

In addition to that, we proposed a hybrid method that employs both techniques presented herein. The hybrid method was shown to provide us with high quality solutions quickly, making it a reliable alternative whenever a good solution is readily required.

We believe that the aforementioned methods can be further investigated to include smart pre–processing and post-processing. Especially in the case of the element partitioning, the algorithm implemented creates random partitions of the same size. This randomization causes the algorithm to be fast, however a smarter selection of the partitions can be looked into to reach solutions of even higher quality.

CHAPTER 4
CLIQUE CENTRALITY

4.1 Preliminaries

Centrality is one of the most fundamental concepts in network analysis; it is a measure of "importance" or "influence" of an actor within a network and is commonly used to identify its most "central" actors. It has been extensively studied in the literature from both theoretical and experimental perspectives starting from the seminal papers of Bavelas [14, 15], Leavitt [85], and Sabidussi [114] in the 1950s, and is primarily considered in social network analysis. Since then, a diverse set of indices has been proposed (see [20, 22, 77] for extended reviews) and applied in various areas including traffic monitoring [40], internet [110], electrical [96], biological [121] and physical [13] networked systems.

The majority of centrality measures can be categorized into three primary classes based on the following concepts [52]: degree, closeness, and betweenness. In each class, the base centrality metric of a node A in a network is defined as follows:

Definition 1. Basic Centrality Measures

- Degree centrality: The number of nodes adjacent to A.
- Closeness centrality: The shortest (maximum shortest, average shortest) distance to node A from any other node in the network.
- Betweenness centrality: The fraction of shortest paths connecting any two other nodes B, C that pass through A.

These three main classes of centrality metrics capture different aspects of "influence" of graph nodes, and depending on the application, each one of these metrics may be preferred over another. Specifically, if the edges in a social network represent direct contacts between two persons, then a person with more direct contacts (high degree) can be viewed as a major communication channel of information or be involved in the main stream of information flow in the network. Conversely, a person with less direct contacts (low degree) is barely involved in the information flow with others and may lack an active

participation in the ongoing communication process. Similarly, a person located on the communication paths that links a large number of people together (high betweenness centrality), may exhibit a potential for influence or control over their communication. The concept of closeness centrality was originally suggested as a measure of potential control of others when a person passes its information through the network [15]; the fewer the number of intermediaries in the communication paths between a person and others, the more central that person is considered to be. For more discussion on the intuition and potential applications of these centrality metrics, we refer the reader to [52].

More recently, Everett and Borgatti [45, 46] extended these three standard network centrality measures to groups and classes. They argue that group centrality measures will enable researchers to analyze the relative influence of groups or teams with certain attributes (age, ethnicity, club membership, or occupation) within an organization. Moreover, in communication or transportation networks, a group of nodes with the highest betweenness centrality may correspond to the best candidate locations for a distributed monitoring system, which can be used for network monitoring tasks such as traffic measurements [40], assuming that node pairs use the shortest paths for communication. However, as opposed to degree and closeness centrality, computing betweenness centrality of any given group is a computationally demanding task. The problem of computing group betweenness centrality, as well as the identification of the most prominent group with the maximum betweenness centrality, are considered in [40, 41, 49, 76, 109].

Alternatively, group centrality measures can be applied to sets of informal groups of individuals forming cohesive subgroups (e.g., cliques), which would allow to analyze which ones are the most and the least central in the network. The clique concept, which was originally introduced by Luce and Perry [87] in 1949 to model a notion of cohesive subgroups in social networks analysis, is a subset of vertices that are pairwise adjacent. It ensures perfect communication between group entities as they are directly linked to each other. Cliques have a variety of practical applications in science and engineering (see [2,

18, 21, 29, 54] and references therein). Not only cliques with maximum centralities, as the most influential clusters, but also cliques with minimum centralities can be of particular interest. For instance, in the context of social networks, cliques with very law centralities may represent highly-connected communities, which have very poor communication ability with the rest of the network. Thus, the members of such communities might be more likely to adhere to the opinions of their group members and less likely to be influenced by information which they receive from other network members, which makes it difficult to reach a social consensus in a whole network. The effect of such "remote" communities on the network consensus characteristics is mentioned in a recent work [44].

In this chapter, we consider the problem of detecting cliques with maximum and minimum centralities, using the aforementioned standard centrality measures. Specifically, we address the following question: which group of k members inducing a clique is the most and the least central in a given network? As most of the problems related to finding cliques are $\mathcal{NP}$-hard, the problems that we consider are also $\mathcal{NP}$-hard. In this work, we develop mixed integer programming formulations for all considered centrality metrics. We demonstrate the performance of the proposed formulations on synthetic and real-life network instances. Interestingly, our findings indicate that the most central cliques in real-life networks are usually smaller than the size of the largest clique in the same graph. This may be partially explained by the fact that larger cliques have less available nodes to influence. Moreover, the number of largest cliques is relatively small in comparison to the number of cliques of with fewer number of nodes.

The chapter proceeds as follows. Section 2 introduces notations and definitions related to three main group centrality measures, formally defines the clique centrality problem and briefly discusses its complexity. In Section 3, we develop linear integer programming formulations for the corresponding optimization problems. Section 4 illustrates the performance of the proposed formulations on various real-life and synthetic network instances. Moreover, we discuss some interesting insights concerning cliques with the

maximum and minimum centralities. Finally, we give our concluding remarks and suggest the directions for future work following the computational results in Section 4.

4.2 Definitions, Notations

Consider a simple undirected graph $G = (V, E)$ with a set of n nodes (vertices) $V = \{1, \ldots, n\}$ and a set of edges $E \subset V \times V$. Two vertices i and j are connected in G if there is a path $\mathcal{P}_{ij}$ between them, i.e, there exists a set of vertices $\{v_0, \ldots, v_\ell\} \subseteq V$ such that $v_0 = i$, $v_\ell = j$, and $(v_k, v_{k+1}) \in E, \forall k = 0, \ldots, \ell - 1$. A graph G is connected if every pair of nodes $i, j \in V$ is connected. For simplicity of exposition, we only consider connected graphs G, although this work can be generalized to disconnected graphs.

A path $\mathcal{P}_{ij}$ between i and j in G is the shortest one (note that it is not necessarily unique), and is called the graph geodesic if it contains the least number of edges among all paths connecting i and j in G. Let $g(i, j)$ be the number of geodesics (distinct shortest paths) between nodes i and j, and $g_t(i, j)$, $g_e(i, j)$ be the number of geodesics that contain node $t \in V$ or edge $e \in E$ ($e \neq (i, j)$, $i \neq t$, $j \neq t$), respectively. The length (i.e., number of edges) of the shortest path between i and j in G is referred to as the distance between i and j in G and is denoted by $d(i, j)$; the largest distance in a graph is referred to as the diameter, $diam(G) = \max_{i,j \in V} d(i, j)$. For any node i, let $N(i) = \{j | (i, j) \in E\}$ be the set of neighbors of node i.

For any subset $S \subseteq V$, the subgraph induced by S in G is defined by $G[S] = (S, (S \times S) \cap E)$. A node belonging to S is referred to as a group node, while nodes in $V \setminus S$ are considered to be the non-group nodes. Let $g_S(i, j)$ be the number of geodesics between nodes $i, j \in V \setminus S$ passing through S; $N(S) = \{j \in V \setminus S | (i, j) \in E, i \in S\}$ be the set of neighbors of S; and for any node $i \in V \setminus S$, let $d(i, S) = \min_{j \in S} d(i, j)$ be the distance between i and S. To be consistent with the existing studies on social networks, in the definitions below, we call a subgraph $G[S]$ as a group S. Following the above notations, the group centralities are defined as follows [45, 46]:

Definition 2. Group Degree Centrality

A group S degree centrality $\mathcal{C}^d(S)$ is the number of non-group nodes that are connected to S:

$$\mathcal{C}^d(S) = |N(S)| \tag{4–1}$$

Note that multiple links to the same node are counted only once.

Definition 3. Group Closeness Centrality

A group S closeness centrality is based on the distances from S to all nodes $V \setminus S$ outside the group. We consider two variants of closeness centrality, the maximum distance $\mathcal{C}_1^c(S)$ and the total distance $\mathcal{C}_2^c(S)$ to outside nodes:

$$\mathcal{C}_1^c(S) = \max_{i \in V \setminus S} d(S, i) \qquad\qquad \mathcal{C}_2^c(S) = \sum_{i \in V \setminus S} d(S, i) \tag{4–2}$$

Observe that since we consider groups with a fixed size, optimizing the total distance is equivalent to optimizing the average distance from a group S to outside nodes $V \setminus S$, which is equal to $\mathcal{C}_2^c(S)/(n-k)$. In the existing literature, the maximum distance from a given node to every other node is known as node eccentricity [64]. In this sense, $\mathcal{C}_1^c(S)$ can be viewed as a group eccentricity metric.

Definition 4. Group Betweenness Centrality

A group S betweenness centrality measure is the proportion of geodesics connecting pairs of non-group members that pass through S:

$$\mathcal{C}^b(S) = \sum_{i,j \in V \setminus S} \frac{g_S(i,j)}{g(i,j)} \tag{4–3}$$

Note that the term $\frac{g_S(i,j)}{g(i,j)}$ in the definition of betweenness centrality in Equation 4–3 is also known in the existing literature [51] as the probability that a pair of nodes $i, j \in V \setminus S$ communicates using a path going through S if they they select a communication path randomly among all possible shortest paths. In this case, $\mathcal{C}^b(S)$ is the average number of communications in the graph G which go through S. In addition, one would naturally

consider the optimistic and the pessimistic cases of betweenness centrality defined as

$$\mathcal{C}^{b^-}(S) = \sum_{i,j \in V \setminus S} \left\lfloor \frac{g_S(i,j)}{g(i,j)} \right\rfloor, \qquad \mathcal{C}^{b^+}(S) = \sum_{i,j \in V \setminus S} \left\lceil \frac{g_S(i,j)}{g(i,j)} \right\rceil \tag{4–4}$$

In the optimistic case, each pair of nodes $i, j \in V \setminus S$ always prefers to communicate using a path going through S if there is a geodesic going through S. In the most pessimistic case, each pair of nodes $i, j \in V \setminus S$ always tries to avoid to communicate through a path going through S; thus, they will communicate through nodes in S if all geodesics go through S.

Observe that, as opposed to standard betweenness centrality $\mathcal{C}^b(\cdot)$, computing $\mathcal{C}^{b^+}(S)$ is straightforward and can be done using centralities $\mathcal{C}^{b^+}(\cdot)$ of individual nodes in S since

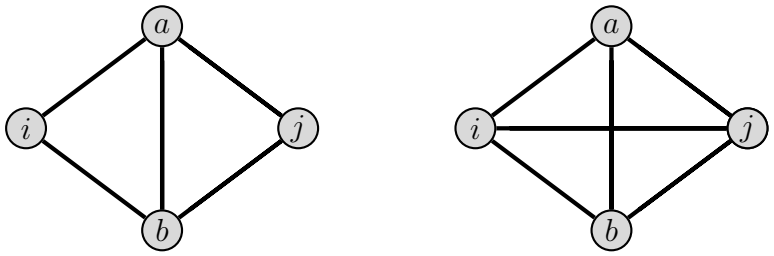
$$\mathcal{C}^{b^+}(S) = \sum_{i,j \in V \setminus S} \max_{t \in S} \left\lceil \frac{g_t(i,j)}{g(i,j)} \right\rceil = \sum_{i,j \in V \setminus S} \max_{t \in S} \mathcal{C}^{b^+}(t) \tag{4–5}$$

However, $\mathcal{C}^{b^-}(S)$ cannot be computed just using $\mathcal{C}^{b^-}(\cdot)$ for individual nodes. For example, if $S = \{a, b\}$ it is easy to construct an example of two graphs in which there exists a pair of nodes i, j such that $\mathcal{C}^{b^-}(a) = \mathcal{C}^{b^-}(b) = 0$ in both graphs, but $\mathcal{C}^{b^-}(S) = 1$ in one graph, and $\mathcal{C}^{b^-}(S) = 0$ in the other (Fig. 4-1).

A clique S is a subset of V such that the subgraph $G[S]$ induced by S in G is complete [21], i.e., all its nodes are pairwise adjacent. The maximum clique problem seeks for a clique S of maximum cardinality $|S|$ in G. The decision version of this problem is $\mathcal{NP}$-complete [56]. In this study, we focus on finding cliques of a given size with maximum and minimum centralities. The formal definition of the problem for finding the most and the least central cliques and its decision version are the following:

Problem 1 (clique with maximum/minimum centrality). Given a graph $G = (V, E)$, a centrality metric $\mathcal{C}(\cdot) \in \{\mathcal{C}^d(\cdot), \mathcal{C}^c_1(\cdot), \mathcal{C}^c_2(\cdot), \mathcal{C}^b(\cdot), \mathcal{C}^{b^-}(\cdot), \mathcal{C}^{b^+}(\cdot)\}$ and a positive integer k, find a clique S of size k ($k = |S|$) with maximum/minimum centrality value $\mathcal{C}(S)$.

Figure 4-1. An illustrative example that computing individual node betweenness centralities is not enough to compute $\mathcal{C}^{b^-}(S)$.



Definition 5. CLIQUE CENTRALITY: Given a graph $G = (V, E)$, a centrality metric $\mathcal{C}(\cdot) \in \{\mathcal{C}^d(\cdot), \mathcal{C}^c_1(\cdot), \mathcal{C}^c_2(\cdot), \mathcal{C}^b(\cdot), \mathcal{C}^{b^-}(\cdot), \mathcal{C}^{b^+}(\cdot)\}$, a non-negative number s and an integer k, is there a clique $S \subseteq V$ of size k such that $\mathcal{C}(S) \geq s$ (or $\mathcal{C}(S) \leq s$)?

As a final remark we note that the CLIQUE CENTRALITY problem is $\mathcal{NP}$-complete since it contains the CLIQUE problem as a special case when $s = 0$ (or $s = |V|^2$). Therefore, Problem 1 is $\mathcal{NP}$-hard.

4.3 Mixed Integer Programming (MIP) Formulations

In this section, we develop linear mixed 0–1 programming formulations for finding cliques with maximum/minimum centralities for all considered centrality metrics: degree $\mathcal{C}^d(\cdot)$, closeness $\mathcal{C}^c_1(\cdot)$, $\mathcal{C}^c_2(\cdot)$, and three versions of betweenness $\mathcal{C}^b(\cdot)$, $\mathcal{C}^{b^-}(\cdot)$, $\mathcal{C}^{b^+}(\cdot)$.

Let x_i ($i \in V$) be a 0-1 variable such that $x_i = 1$ iff $i \in V$ is in a clique S. To model clique constraints we use the standard technique that if there is no edge between nodes i and j, i.e., $(i, j) \notin E$, then a clique S cannot simultaneously contain both nodes i and j:

$$x_i + x_j \leq 1, \quad \forall (i, j) \notin E. \tag{4–6}$$

Below are the modeling techniques and IP formulations for each centrality metric.

4.3.1 Degree Centrality

The IP formulation for degree centrality is straightforward. Let $y_i = 1$ iff $i \in N(S)$. Then, $\mathcal{C}^d(S) = \sum_{i \in V} y_i$ and the formulation can be written as follows:

$$(\text{MAX-}\mathcal{C}^d) : \max \sum_{i \in V} y_i \tag{4–7}$$

$$s.t.$$

$$y_i \leq \sum_{j:(i,j) \in E} x_j, \qquad \forall i \in V \tag{4–8}$$

$$x_i + y_i \leq 1, \qquad \forall i \in V \tag{4–9}$$

$$x_i + x_j \leq 1, \qquad \forall (i,j) \notin E \tag{4–10}$$

$$\sum_{i \in V} x_i = k, \tag{4–11}$$

$$x_i, y_i \in \{0,1\}, \qquad \forall i \in V \tag{4–12}$$

where Constraints 4–8 ensure that $y_i = 0$ if i is not a neighbor of a clique S ($i \notin N(S)$), Constraints 4–9 ensure that $y_i = 0$ if i is in a clique S. Note that since this is a maximization problem, we do not need an extra set of constraints to enforce $y_i = 1$ for all other cases. However, these constraints will be required for the minimization version of the problem. The remaining set of Constraints 4–10 and 4–11 enforce S to be a clique of size k.

Similarly, we obtain the formulation for finding a clique of size k with minimum $\mathcal{C}^d(S)$:

$$(\text{MIN-}\mathcal{C}^d) : \min \sum_{i=1}^{|V|} y_i \tag{4–13}$$

$$s.t.$$

4–8 throughout 4–11,

$$x_i + y_i \geq \frac{1}{\mu_i} \sum_{j:(i,j)\in E} x_j, \qquad \forall i \in V \tag{4–14}$$

$$x_i, y_i \in \{0,1\}, \qquad \forall i \in V \tag{4–15}$$

where Constraints 4–14 ensure that $y_i = 1$ if i is not in a clique S and is a neighbor of at least one clique node, μ_i is sufficiently large constant, which for the tightness of LP relaxation purposes can be set to $\mu_i = \min(|N(i)|, k)$. Note that Constraints 4–8 and 4–9 can be omitted due to the structure of the objective function.

4.3.2 Closeness Centrality

To calculate the closeness centrality $\mathcal{C}_1^c(S)$ or $\mathcal{C}_2^c(S)$ of a clique S, we need to compute the distance from every node $i \in V \setminus S$ to the clique S. To model such distances in the corresponding IP formulations, we introduce a set of binary variables $x_i^{(\ell)}$ $(i \in V, \ell = 1, \ldots, diam(G))$ which is defined as follows. First, the variables $x_i^{(0)}$ $(i \in V)$ determine the clique S, i.e, $x_i^{(0)} = x_i = 1$ iff $i \in S$. Then, the variables $x_i^{(1)}$ $(i \in V)$ define the set of nodes that is either in S or adjacent to S, i.e., $x_i^{(1)} = 1$ iff $i \in N(S) \cup S$. For other values ℓ, $x_i^{(\ell)} = 1$ iff $d(i,S) \leq \ell$. Observe that the number of nodes, whose distance is ℓ from a clique S is $\sum_{i\in V} x_i^{(\ell)} - \sum_{i\in V} x_i^{(\ell-1)}$. Since $d(i,S) \leq diam(G)$ for any clique S and $i \in V \setminus S$, then we just need to consider $\ell = 1, \ldots, diam(G)$.

The appropriate values of these variables in the corresponding IP formulations are modeled recursively using the following observation: $x_i^{(\ell)} = 1$ iff $x_i^{(\ell-1)} = 1$, or i is adjacent to node j such that $x_j^{(\ell-1)} = 1$. This observation can be written in terms of linear

constraints as follows:

$$x_i^{(\ell)} \leq \sum_{j:(i,j)\in E} x_j^{(\ell-1)} + x_i^{(\ell-1)}, \qquad \forall i \in V,\, \ell = 1, \ldots, diam(G) \tag{4–16}$$

$$x_i^{(\ell)} \geq \frac{1}{\mu_i}\left(x_i^{(\ell-1)} + \sum_{j:(i,j)\in E} x_j^{(\ell-1)}\right), \qquad \forall i \in V,\, \ell = 1, \ldots, diam(G) \tag{4–17}$$

where μ_i is sufficiently large constant, which for the tightness of LP relaxation purposes can be set to $\mu_i = 1 + |N(i)|$.

Recall that we consider two versions of closeness centrality metrics $\mathcal{C}_1^c(S)$ and $\mathcal{C}_2^c(S)$ which consider the maximum and the total (or average) distance to a clique S, respectively. For the first metric, $\mathcal{C}_1^c(S)$, we introduce an extra set of binary variables v_ℓ ($\ell = 1, \ldots, diam(G)$), where v_ℓ is equal to 1 iff there is at least one node with distance ℓ to S. In other words, $v_\ell = 1$ iff $\sum_{i\in V} x_i^{(\ell)} - \sum_{i\in V} x_i^{(\ell-1)} \geq 1$. In this case, $\mathcal{C}_1^c(S)$ can be expressed as $\mathcal{C}_1^c(S) = \sum_{\ell=1}^{diam(G)} v_\ell$, which leads to the following formulation for finding a clique S with the minimum distance to the outside nodes:

$$(\text{MIN-}\mathcal{C}_1^c) : \min \sum_{\ell=1}^{diam(G)} v_\ell \tag{4–18}$$

$$s.t.$$

4–10, 4–11, 4–16, 4–17

$$v_\ell \leq \sum_{j\in V}(x_j^{(\ell)} - x_j^{(\ell-1)}), \qquad 1 \leq \ell \leq diam(G) \tag{4–19}$$

$$v_\ell \geq \frac{1}{|V|}\sum_{j\in V}(x_j^{(\ell)} - x_j^{(\ell-1)}), \qquad 1 \leq l \leq diam(G) \tag{4–20}$$

$$x_i^{(\ell)}, v_\ell \in \{0,1\}, \qquad \forall i \in V, 0 \leq \ell \leq diam(G)$$

Constraints 4–19 and 4–20 ensure that variable $v_\ell = 1$ if and only if there exists at least one node with distance ℓ from the clique S. For the centrality metric $\mathcal{C}_2^c(\cdot)$ which counts the total distance, the formulation is straightforward:

$$(\text{MIN-}\mathcal{C}_2^c): \min \sum_{\ell=1}^{diam(G)} \ell \left(\sum_{i \in V} x_i^{(\ell)} - \sum_{i \in V} x_i^{(\ell-1)} \right) \tag{4–21}$$

$$s.t.$$

4–10, 4–11, 4–16, 4–17

$$x_i^{(\ell)} \in \{0,1\} \qquad \forall i \in V, 0 \leq \ell \leq diam(G)$$

Observe that both formulations MIN-$\mathcal{C}_1^c$ and MIN-$\mathcal{C}_2^c$ identify the most central cliques since they find cliques with minimum distance or total distance to other nodes. The formulations contain $\Theta(n \times diam(G))$ variables and constraints. In order to find the least central cliques, the objectives in the corresponding formulations have to be maximized. Note that this technique can be also used for developing formulations with other versions of closeness centrality metrics, such as $\mathcal{C}^c(S) = \sum_{i \in V \setminus S} \frac{1}{d(i,S)}$ or $\mathcal{C}^c(S) = \sum_{i \in V \setminus S} \frac{1}{2^{d(i,S)}}$ [36].

4.3.3 Betweenness Centrality

4.3.3.1 Standard "Probabilistic" Case

As we mentioned before, computing standard betweenness centrality defined by Equation 4–3 for a group is not an easy task. However, when the group forms a clique, its betweenness centrality can be computed using centralities of individual nodes and edges, as we demonstrate in the following proposition.

Proposition 4.1. Let $S \subset V$ be a clique in a graph $G = (V, E)$ with the set of edges $E_s = (S \times S) \cap E$. Then, for any pair of nodes $i, j \in V \setminus S$:

$$g_S(i,j) = \sum_{t \in S} g_t(i,j) - \sum_{e \in E_s} g_e(i,j). \tag{4–22}$$

Proof. Observe that any geodesic path $\mathcal{P}_{ij}$ between nodes i and j which goes through the clique S may contain one or two clique nodes only. Otherwise, it is easy to verify that it cannot be geodesic, i.e., the shortest one.

The number of geodesic paths going through only one clique node t for all $t \in S$ is appropriately counted in the expression $\sum_{t \in S} g_t(i,j)$. However, the number of geodesic paths going through two clique nodes s,t in the expression $\sum_{t \in S} g_t(i,j)$ is counted twice for any possible $s,t \in S$ (once for s and once for t). Observe that if a shortest path $\mathcal{P}_{ij}$ contains nodes $s,t \in S$ then it goes through an edge $(s,t) \in E_s$ as the clique S contains all possible edges. Hence, to get the right number of total geodesic paths between nodes i and j going through a clique S, we need to substract $\sum_{e \in E_s} g_e(i,j)$ from $\sum_{t \in S} g_t(i,j)$, which ends the proof of the proposition. □

For the IP formulation, we need to compute $g_t(i,j)$ and $g_e(i,j)$ for all $i,j,t \in V$ and $e \in E$. For simplicity, let $g^{ij} = g(i,j)$, $g_t(i,j) = g_t^{ij}$ and $g_e(i,j) = g_e^{ij}$, where an index t refers to nodes and e refers to edges. Also, let u_e ($e \in E$) be a binary variable such that $u_e = 1$ iff $e \in E_s$. To calculate g_t^{ij}, we use the classical all-pairs shortest path algorithm of Floyd-Warshall [50], where we compute all shortest paths between any two nodes of the graph. The algorithm was modified to account for multiple shortest paths connecting two nodes. For every edge $e \in E$, $g_e(i,j)$ is a byproduct of the previous calculations.

The formulation for finding a clique with the maximum standard betweenness centrality can be written as follows:

$$(\text{MAX-}\mathcal{C}^b) : \max \sum_{i,j \in V} \left(\frac{\sum_{t \in V} g_t^{ij} x_t - \sum_{e \in E} g_e^{ij} u_e}{g^{ij}} \right) \tag{4–23}$$

$$s.t.$$

4–10, 4–11,

$$u_e \leq x_i, u_e \leq x_j, u_e \geq x_i + x_j - 1, \qquad \forall e = (i,j) \in E \tag{4–24}$$

$$x_i, u_e \in \{0,1\}, \qquad \forall i \in V, e \in E \tag{4–25}$$

where Constraint 4–24 ensure that $u_e = 1$ if and only if $e \in E_c$.

Observe that the objective function in Equation 4–23 also includes the geodesics between clique and non-clique members. The next proposition shows that it still returns the correct value of betweenness centrality defined by Equation 4–3.

Proposition 4.2. The objective function of Equation 4–23 returns the correct value for betweenness centrality, i.e.,

$$\mathcal{C}^b(S) = \sum_{i,j \in V \setminus S} \frac{g_S(i,j)}{g(i,j)} = \sum_{i,j \in V} \left(\frac{\sum_{t \in V} g_t^{ij} x_t - \sum_{e \in E} g_e^{ij} u_e}{g^{ij}} \right) \tag{4–26}$$

Proof. By Proposition 4.1 and definition of variables x_t and u_e,

$$\mathcal{C}^b(S) = \sum_{i,j \in V \setminus S} \frac{g_S(i,j)}{g(i,j)} = \sum_{i,j \in V \setminus S} \left(\frac{\sum_{t \in V} g_t^{ij} x_t - \sum_{e \in E} g_e^{ij} u_e}{g^{ij}} \right)$$

Hence, to prove the proposition, we need to show that

$$\sum_{t \in S} g_t^{ij} - \sum_{e \in E_s} g_e^{ij} = 0, \quad \forall i \in S, j \in V. \tag{4–27}$$

Consider two possible cases for node j separately:

- $j \in S$: since S is a clique, then there is only one shortest path between nodes i and j $(i, j \in S)$ and it goes though an edge $(i, j) \in E_s$. Hence, $g_e^{ij} = 0$ for all $e \in E_s$ and $g_t^{ij} = 0$ for all $t \in S$, and Equation 4–27 is true for $i, j \in S$.
- $j \in V \setminus S$: if a geodesic $\mathcal{P}_{ij}$ from a clique node i to the non-clique node j does not contain clique nodes, then $g_e^{ij} = 0$ for all $e \in E_s$ and $g_t^{ij} = 0$ for all $t \in S$, and Equation 4–27 is also true. However, if $\mathcal{P}_{ij}$ goes through a clique S, it must have go through only one other node $t \in S$ and an edge $(i, t) \in E_s$. Otherwise, the shortest path definition will be violated. In this case, the number of such paths will be counted in $\sum_{t \in S} g_t(i,j)$ and in $\sum_{e \in E_s} g_e(i,j)$ only once and Equation 4–27 still holds.

□

4.3.3.2 Pessimistic Case

We introduce new 0-1 variables t_{ij} $(i, j \in V)$ such that $t_{ij} = 1$ iff all geodesics between nodes i, j pass through a clique S. Then, the problem formulation which maximizes $\mathcal{C}^{b^-}(\cdot)$ becomes the following:

$$(\text{MAX-}\mathcal{C}^{b^-}) : \max \sum_{i,j \in V} t_{ij} \tag{4–28}$$

$$s.t.$$

4–10, 4–11, 4–24,

$$g^{ij} t_{ij} \leq \sum_{t \in V} g_t^{ij} x_t - \sum_{e \in E} g_e^{ij} u_e, \qquad \forall i, j \in V \tag{4–29}$$

$$t_{ij} + x_i \leq 1, t_{ij} + x_j \leq 1, \qquad \forall i, j \in V \tag{4–30}$$

$$x_i, t_{ij}, u_e \in \{0, 1\}, \qquad \forall i, j \in V, e \in E \tag{4–31}$$

where Constraints 4–29 ensure that $t_{ij} = 0$ iff at least one geodesic between nodes i, j does not pass through a clique S, and Constraints 4–30 make sure that only geodesics between non-clique nodes are considered.

The formulation for minimizing betweenness $\mathcal{C}^{b^-}(\cdot)$ is the following:

$$(\text{MIN-}\mathcal{C}^{b^-}) : \min \sum_{i,j \in V} t_{ij} \tag{4–32}$$

$$s.t.$$

4–10, 4–11, 4–24,

$$t_{ij} \geq 1 + \sum_{t \in V} g_t^{ij} x_t - \sum_{e \in E} g_e^{ij} u_e - g^{ij} - x_i - x_j, \qquad \forall i, j \in V \tag{4–33}$$

$$x_i, t_{ij}, u_e \in \{0, 1\}, \qquad \forall i, j \in V, e \in E \tag{4–34}$$

where Constraints 4–33 ensure that $t_{ij} = 1$ iff all geodesics between nodes $i, j \in V \setminus S$ pass through a clique S. Note that Constraints 4–29 and 4–30 are not required due to the structure of the objective function.

4.3.3.3 Optimistic Case

For the optimistic case, recall that the betweenness centrality can be computed using individual node centralities as

$$\mathcal{C}^{b^+}(S) = \sum_{i,j \in V \setminus S} \max_{t \in S} \left\lceil \frac{g_t(i,j)}{g(i,j)} \right\rceil \tag{4–35}$$

Let z_{ij} $(i, j \in V)$ be a binary variable such that $z_{ij} = 1$ iff there exists a shortest path between nodes i and j which passes through at least one node in the clique S. Let also,

$$f_t^{ij} = \left\lceil \frac{g_t(i,j)}{g(i,j)} \right\rceil, \forall i, j, t \in V \tag{4–36}$$

Then, the problem formulation for finding a clique with the maximum $\mathcal{C}^{b^+}(\cdot)$ is the following:

$$(\text{MAX-}\mathcal{C}^{b^+}) : \max \sum_{i,j \in V} z_{ij} \tag{4–37}$$

$$s.t.$$

4–10, 4–11,

$$z_{ij} \geq \frac{1}{|V|} \sum_{t \in V} f_t^{ij} x_t - x_i - x_j, \qquad \forall i, j \in V \tag{4–38}$$

$$z_{ij} \leq \sum_{t \in V} f_t^{ij} x_t \qquad \forall i, j \in V \tag{4–39}$$

$$x_i, z_{ij} \in \{0, 1\}, \qquad \forall i, j \in V \tag{4–40}$$

In the objective function in Equation 4–37, we aim to maximize the number of shortest paths that use the nodes of the clique. Constraints 4–38 and 4–39 model the indicator variables z_{ij}, by enforcing them to be equal to 1 whenever there exists a shortest path between nodes i and j that passes through a node of the clique S, and 0 otherwise. To obtain the formulation for finding the least central cliques with the minimum value of betweenness centrality, the objective needs to be minimized with the

same set of constraints. Note also that Constraints 4–38 and 4–39 can be omitted for the maximization/minimization version of the problem.

4.4 Computational Experiments

In this section, we present the results of our computational experiments to demonstrate the performance of the developed MIP formulations and numerically illustrate the difference between maximum and minimum clique centralities with respect to the clique size $|S| = k$ in real-life and synthetic networks using various centrality metrics.

The computational experiments were performed on a server with two AMD Opteron 6128 Eight-Core CPUs and 12 GB of RAM, running Linux x86 64, CentOS 5.9. All formulations were implemented in C++ and solved using Gurobi 5.50, while graph operations were performed using NetworkX [63].

The real-life network instances are obtained from publicly available data from the University of Florida Sparse Matrix Collection database [37] (which also includes DIMACS10 Challenge Collection [38]) and COLOR02/03/04 [1]. Specifically, the considered networks include: karate (social network of a karate club with 34 members at a US university in the 1970s [123]), krebs (terrorist network of the 9/11 hijackers and their associates, compiled by Krebs [81]), dolphins (social network of frequent associations between 62 dolphins in a community living off Doubtful Sound, New Zealand [88]), ieeebus (the IEEE 118 Bus Test Case is a portion of the American Electric Power System in the Midwestern US as of December, 1962 [37]), and book graphs anna, david, huck, jean corresponding to four classic works: Tolstoy's Anna Karenina, Dicken's David Copperfield, Twain's Huckleberry Finn, and Hugo's Les Misérables [1]. We also include experiments on two classes of randomly generated graph instances: Erdos-Renyi and Barabási-Albert graphs generated according to the classical $G(n, p)$ [42], and preferential attachments models [112], respectively.

The results of the computational experiments are summarized in Tables 4-1-4-4. For any graph instance, we consider all possible values k for which a clique of size k

exists in this graph and identified cliques with maximum and minimum centralities $\mathcal{C}^d(\cdot), \mathcal{C}_1^c(\cdot), \mathcal{C}_2^c(\cdot)$, $\mathcal{C}^b(\cdot)$, $\mathcal{C}^{b^-}(\cdot)$, $\mathcal{C}^{b^+}(\cdot)$. For a better representation, the values of degree and betweenness centralities are reported as a percentage of maximum possible values, i.e., we scaled them by $n-1$ and $(n-k)(n-k-1)/2$, respectively. We also scale $\mathcal{C}_2^c(\cdot)$ by $n-k$ so that the corresponding value represent the average distance from a clique to outside nodes.

One of the immediate observations is that the variation between maximum and minimum centralities of the same size cliques is quite large even for the cliques whose sizes are close to the size of the maximum clique. Moreover, for several real-life network instances (krebs, dolphins, ieeebus and some book graphs), the cliques with the maximum centralities among all possible clique sizes are not necessarily the largest ones. However, no such pattern is observed in synthetic networks, which may be attributed to their specific topological structure. These observations suggest that large cliques are not necessarily the most central in real-life networks. In addition, the centrality values of most central cliques seem to be reasonably large too. For example, there is a clique of size 3 in krebs network such that its neighbor set contains more than 50% of the remaining nodes. Also, there is a clique of size 3 in dolphins network such that more than 80% pairs of nodes have at least one shortest path going through this clique. Note that the individual node centralities are not that large in the corresponding experiments.

Table 4-2 provides the computational times for solving problems presented in Table 4-1. Note that the computational runtime does not always increase with the size of the cliques that are being considered. Instead, there is a consistent spike in "average-sized" cliques. Further, we can observe that computing betweenness centrality using the optimistic model usually takes considerably more time than using the standard (probabilistic) version of the metric, or its pessimistic counterpart.

In this chapter, we considered a centrality-based model to measure the relative influence of cliques within a network by utilizing three classical group centrality measures: degree, closeness, and betweenness as well as their intuitive alternations. For each

centrality metric we developed a linear 0-1 programming formulation for identifying cliques of any given size k with maximum and minimum centrality values. The numerical experiments demonstrate that large cliques may exhibit a significant variation in their connection to the rest of the network. Moreover, the most central cliques are not usually the largest ones, which suggests that the perfect communication among clique members does not necessarily imply their good integration within a whole network. The directions of future research may include the developments of more advanced exact or heuristic methods to handle large-scale network instances. In addition, as cliques are sometimes viewed as overly restrictive group structures, this approach can be generalized for studying centralities of less restrictive but still highly-connected subgraphs known as clique-relaxations [101], i.e., quasi-cliques, k-plexes, k-clubs, etc.

Table 4-1. Maximum and minimum clique centralities in real-life social and power grid network instances.

$\|S\|$	Degree (%) $\mathcal{C}^d()$		Closeness $\mathcal{C}_1^c()$		$\mathcal{C}_2^c()$		$\mathcal{C}^b()$		Betweenness (%) $\mathcal{C}^{b^-}()$		$\mathcal{C}^{b^+}()$	
	Max	Min	Min	Max	Min	Max	Max	Min	Max	Min	Max	Min
karate: $\|V\| = 34, \|E\| = 78$, max clique size = 5												
1	51.51	3.12	3	5	1.76	3.52	39.39	3.13	16.97	0.00	54.19	5.88
2	62.50	9.38	2	5	1.38	2.66	51.69	4.36	21.16	0.00	66.13	11.59
3	64.52	9.68	2	4	1.35	2.68	59.71	7.96	23.37	0.00	68.98	17.11
4	63.33	40.00	2	4	1.37	1.80	67.20	10.62	24.79	1.38	70.94	39.48
5	62.07	58.62	2	3	1.38	1.45	67.74	15.24	24.95	2.59	79.76	69.16
krebs: $\|V\| = 62, \|E\| = 153$, max clique size = 6												
1	36.07	1.64	3	5	1.70	4.00	49.34	1.67	30.36	0.00	72.71	3.23
2	51.67	3.33	2	5	1.55	3.25	61.13	2.00	35.28	0.00	77.63	6.40
3	55.93	6.78	2	5	1.51	3.08	68.38	3.95	37.89	0.00	81.17	9.52
4	48.27	12.07	3	4	1.59	2.48	72.61	6.20	38.91	0.21	81.86	14.67
5	47.37	15.79	3	4	1.60	2.37	73.08	11.34	38.96	1.03	81.76	23.51
6	42.86	30.36	3	4	1.64	2.18	73.08	15.56	38.68	3.32	80.12	32.05
dolphins: $\|V\| = 62, \|E\| = 159$, max clique size = 5												
1	19.67	1.64	5	8	2.39	5.61	16.71	1.67	9.08	0.00	37.54	3.23
2	30.00	3.33	4	8	2.13	4.68	33.00	2.06	13.79	0.00	49.34	6.40
3	33.90	5.08	4	7	2.02	3.76	36.70	4.55	15.50	0.00	55.63	10.34
4	31.03	13.79	5	7	2.17	3.38	33.00	9.11	13.70	0.88	51.08	16.23
5	28.07	21.05	6	7	2.72	3.25	35.27	16.58	11.95	5.95	37.12	27.74
ieeebus: $\|V\| = 118, \|E\| = 179$, max clique size = 4												
1	7.69	0.85	7	14	4.25	8.85	27.15	0.86	14.16	0.00	40.56	1.69
2	10.34	0.86	7	14	3.97	8.84	44.94	1.19	21.04	0.00	54.21	3.38
3	10.43	2.61	8	12	3.98	7.03	37.17	2.26	18.08	0.00	49.79	5.85
4	6.14	6.14	9	9	5.36	5.36	8.62	5.05	4.12	1.03	10.55	10.55

Table 4-2. Computational times (in seconds) for solving the maximum and minimum clique centrality problems in real-life social and power grid network instances from Table 4-1.

$\|S\|$	Degree (%) $\mathcal{C}^d()$		Closeness $\mathcal{C}_1^c()$		$\mathcal{C}_2^c()$		$\mathcal{C}^b()$		$\mathcal{C}^{b^-}()$		Betweenness (%) $\mathcal{C}^{b^+}()$	
	Max	Min	Min	Max	Min	Max	Max	Min	Max	Min	Max	Min
karate: $\|V\| = 34, \|E\| = 78$, max clique size = 5												
1	0.02	0.03	0.11	0.04	0.06	0.05	0.10	0.12	0.01	0.01	0.27	0.03
2	0.03	0.04	0.25	0.10	0.15	0.09	0.61	0.78	0.04	0.02	1.13	0.58
3	0.65	0.11	0.44	0.43	0.16	0.10	0.58	0.04	0.03	0.02	1.24	1.01
4	0.64	0.10	0.33	0.27	0.13	0.10	0.11	0.10	0.02	0.01	0.98	0.86
5	0.60	0.09	0.30	0.26	0.13	0.10	0.06	0.09	0.02	0.01	0.45	0.37
krebs: $\|V\| = 62, \|E\| = 153$, max clique size = 6												
1	0.05	0.03	0.16	0.17	0.12	0.10	0.27	0.05	0.03	0.01	0.45	0.08
2	0.16	0.16	0.17	0.20	0.25	0.30	1.44	5.28	0.36	0.02	0.67	1.30
3	0.15	0.16	0.24	0.25	0.41	0.52	1.21	0.37	0.28	0.02	1.55	2.17
4	0.14	0.09	0.13	0.13	0.13	0.16	1.73	0.60	0.10	0.01	2.20	1.70
5	0.05	0.05	0.08	0.08	0.08	0.08	0.50	0.27	0.10	0.01	0.68	0.47
6	0.04	0.04	0.04	0.04	0.07	0.06	0.25	0.12	0.08	0.01	0.83	0.31
dolphins: $\|V\| = 62, \|E\| = 159$, max clique size = 5												
1	0.05	0.03	0.23	0.22	0.27	0.17	0.42	0.03	0.03	0.01	1.09	0.06
2	0.14	0.07	1.87	0.40	0.60	0.21	4.07	2.47	0.46	0.02	1.92	1.10
3	0.31	0.12	0.21	0.23	0.77	0.26	8.79	6.88	0.47	0.02	4.74	1.88
4	0.33	0.26	0.41	0.26	0.94	0.34	7.60	0.79	0.48	0.01	19.42	2.24
5	0.06	0.05	0.12	0.11	0.13	0.13	3.42	0.17	0.07	0.01	17.53	0.66
ieeebus: $\|V\| = 118, \|E\| = 179$, max clique size = 4												
1	0.24	0.27	0.31	0.32	10.91	11.28	5.10	0.35	0.19	0.01	5.91	0.56
2	0.92	1.03	22.72	21.95	26.95	17.17	28.64	31.04	1.74	0.03	354.70	343.09
3	0.58	0.59	22.83	14.47	13.04	7.53	30.02	2.98	0.56	0.02	272.58	234.28
4	0.38	0.51	10.86	5.21	4.78	4.35	14.53	1.88	0.42	0.01	171.83	100.29

Table 4-3. Maximum and minimum clique centralities in book graphs.

$\|S\|$	Degree (%) $\mathcal{C}^d()$		Closeness $\mathcal{C}_1^c()$		$\mathcal{C}_2^c()$		$\mathcal{C}^b()$		Betweenness (%) $\mathcal{C}^{b^-}()$		$\mathcal{C}^{b^+}()$	
	Max	Min	Min	Max	Min	Max	Max	Min	Max	Min	Max	Min
huck: $\|V\| = 74, \|E\| = 303$, max clique size = 11												
1	72.60	1.37	3	6	1.38	4.89	56.94	1.39	33.46	0.00	77.67	2.70
2	79.17	1.38	3	6	1.31	4.90	69.79	1.44	38.63	0.00	83.89	5.37
3	83.10	1.41	3	6	1.28	3.96	78.64	2.98	41.58	0.00	86.30	8.00
4	85.71	7.14	3	5	1.26	2.27	81.49	3.47	43.07	0.00	88.26	10.59
5	88.41	7.79	3	5	1.23	2.27	84.93	4.83	44.15	0.00	89.56	13.14
6	88.24	4.42	3	5	1.24	2.29	87.34	6.49	44.94	0.00	90.41	15.66
7	88.06	4.48	3	5	1.24	2.31	87.34	9.72	44.94	0.00	90.41	18.14
8	80.30	4.55	3	5	1.32	2.33	78.53	12.13	41.70	0.00	87.23	20.58
9	80.00	15.38	3	5	1.32	2.09	78.56	15.20	41.71	0.05	87.26	24.86
10	79.69	17.19	4	5	1.33	2.07	78.56	20.14	41.71	1.88	87.26	28.38
11	73.02	73.02	4	4	1.40	1.40	70.12	44.97	38.74	13.37	83.19	83.19
jean: $\|V\| = 80, \|E\| = 257$, max clique size = 10												
1	45.57	1.27	3	6	1.61	4.22	51.30	1.28	26.36	0.00	71.65	2.5
2	61.55	1.28	3	5	1.42	3.26	60.51	1.35	30.83	0.00	80.63	4.97
3	67.53	5.19	3	5	1.36	3.06	68.00	2.68	34.33	0.00	86.30	7.41
4	69.74	3.95	3	5	1.34	3.09	73.16	3.14	36.03	0.00	88.96	9.81
5	66.67	2.67	3	5	1.37	3.12	73.54	4.23	35.31	0.00	87.34	12.18
6	64.86	2.70	3	5	1.39	3.15	73.54	4.75	34.81	0.00	84.21	14.53
7	64.38	4.11	3	5	1.40	2.78	73.99	6.55	34.90	0.00	84.30	16.84
8	34.72	9.72	4	4	1.86	2.26	40.44	7.72	19.15	0.02	49.72	19.98
9	33.80	11.27	4	4	1.87	2.25	40.57	9.28	19.19	0.04	49.81	25.24
10	32.86	21.43	4	4	1.89	2.13	40.66	12.22	19.22	0.58	49.87	36.57
david: $\|V\| = 87, \|E\| = 406$, max clique size = 11												
1	95.35	1.16	2	3	1.05	2.53	57.58	1.18	38.13	0.00	88.24	2.30
2	100	1.18	1	3	1	2.55	64.50	1.20	42.63	0.00	92.19	4.57
3	100	4.76	1	3	1	2.55	69.07	2.45	44.83	0.00	92.97	6.82
4	100	3.61	1	3	1	2.57	73.56	3.60	46.65	0.00	93.69	9.04
5	100	6.10	1	3	1	1.94	79.34	4.56	48.35	0.00	94.33	11.23
6	100	4.93	1	2	1	1.95	82.01	6.95	49.16	0.00	94.76	13.39
7	100	15.00	1	2	1	1.85	83.40	8.72	49.60	0.00	95.11	15.75
8	100	26.58	1	2	1	1.73	83.67	10.44	49.87	0.00	95.35	18.47
9	100	32.05	1	2	1	1.68	84.18	12.17	50.06	0.00	95.54	21.72
10	100	64.94	1	2	1	1.35	84.87	19.08	50.23	0.89	95.70	38.60
11	100	100	1	1	1	1	84.90	29.23	50.25	15.51	95.75	95.72
anna: $\|V\| = 138, \|E\| = 493$, max clique size = 11												
1	51.82	0.73	3	5	1.53	4.24	28.01	0.74	13.66	0.00	55.33	1.45
2	72.79	0.74	3	5	1.28	3.26	41.72	0.75	19.77	0.00	73.91	2.89
3	81.48	2.96	2	4	1.19	2.69	53.72	1.56	24.91	0.00	83.29	4.32
4	87.31	2.99	2	4	1.13	2.69	68.62	2.33	29.31	0.00	89.87	5.73
5	91.73	3.76	3	4	1.09	2.47	75.37	3.13	31.66	0.00	94.20	7.15
6	93.94	5.30	3	4	1.07	2.33	82.12	4.04	33.18	0.00	96.42	8.71
7	94.66	28.24	3	4	1.06	1.74	86.91	7.98	33.90	0.88	97.23	16.76
8	88.46	42.31	3	4	1.12	1.60	77.76	10.43	31.74	1.26	92.82	21.56
9	88.37	51.16	3	4	1.12	1.50	78.42	15.77	31.82	3.04	92.89	29.90
10	76.56	59.38	3	4	1.25	1.42	68.34	23.39	28.37	7.84	84.17	45.65
11	76.38	76.38	3	3	1.25	1.25	68.34	38.15	28.37	21.81	84.17	84.17

Table 4-4. Maximum and minimum clique centralities in randomly generated network instances according to Erdos-Renyi $G(n,p)$ preferential attachments model.

$\|S\|$	Degree (%) $\mathcal{C}^d()$		Closeness $\mathcal{C}_1^c()$		Closeness $\mathcal{C}_2^c()$		Betweenness (%) $\mathcal{C}^b()$		Betweenness (%) $\mathcal{C}^{b^-}()$		Betweenness (%) $\mathcal{C}^{b^+}()$	
	Max	Min	Min	Max	Min	Max	Max	Min	Max	Min	Max	Min
Erdos-Renyi ($n=50, p=0.20$): $\|E\|=249$, max clique size = 4												
1	32.65	10.20	2	3	1.67	2.22	2.89	2.20	2.67	0.00	17.96	4.47
2	52.08	14.58	2	3	1.48	2.06	5.67	3.96	4.57	0.00	28.57	8.05
3	68.09	31.91	2	3	1.32	1.72	8.42	7.13	6.27	0.05	36.16	12.37
4	76.09	43.48	2	2	1.24	1.57	10.78	10.58	7.33	0.29	41.23	18.57
Erdos-Renyi ($n=50, p=0.30$): $\|E\|=375$, max clique size = 4												
1	51.02	16.33	2	3	1.49	1.90	2.22	2.21	1.42	0.00	22.37	4.04
2	75.07	27.08	2	3	1.25	1.73	4.56	4.18	1.66	0.00	32.82	8.16
3	87.23	44.68	2	2	1.13	1.55	9.56	9.21	2.25	0.00	39.91	12.12
4	93.48	60.87	2	2	1.07	1.39	12.70	12.02	6.52	0.05	45.88	16.45
Erdos-Renyi ($n=75, p=0.20$): $\|E\|=528$, max clique size = 4												
1	29.73	9.46	2	3	1.73	2.05	1.70	1.49	1.39	0.00	11.78	2.79
2	50.68	17.81	2	3	1.49	1.85	3.42	2.81	2.69	0.00	20.43	5.53
3	61.11	33.33	2	3	1.39	1.68	5.11	4.68	3.73	0.04	25.66	8.76
4	64.79	46.48	2	2	1.35	1.54	6.95	6.80	4.54	0.12	28.18	11.69
Erdos-Renyi ($n=75, p=0.30$): $\|E\|=835$, max clique size = 6												
1	40.54	17.57	2	3	1.58	1.85	1.49	1.41	0.89	0.00	13.55	2.67
2	67.12	32.88	2	2	1.33	1.67	2.87	2.83	1.72	0.00	24.07	5.30
3	80.56	47.22	2	3	1.19	1.53	4.42	4.38	2.49	0.00	31.64	7.98
4	87.32	63.38	2	2	1.13	1.37	6.17	5.93	3.18	0.00	37.33	10.65
5	90.13	72.86	2	2	1.10	1.27	8.00	7.56	3.79	0.00	40.54	13.53
6	91.30	85.51	2	2	1.09	1.14	10.35	9.21	4.34	0.10	43.75	17.05

Table 4-5. Maximum and minimum clique centralities in randomly generated network instances according to Barabási-Albert preferential attachments model.

$\|S\|$	Degree (%) $\mathcal{C}^d()$		Closeness $\mathcal{C}_1^c()$		Closeness $\mathcal{C}_2^c()$		Betweenness (%) $\mathcal{C}^b()$		Betweenness (%) $\mathcal{C}^{b^-}()$		Betweenness (%) $\mathcal{C}^{b^+}()$	
	Max	Min	Min	Max	Min	Max	Max	Min	Max	Min	Max	Min
Barabási-Albert ($n = 50, \gamma = 2$): $\|E\| = 96$, max clique size = 3												
1	28.57	4.08	3	5	1.84	3.37	18.84	0.67	12.24	0.00	47.59	4.04
2	41.67	6.25	3	5	1.65	3.01	25.71	4.09	18.83	0.00	59.10	8.29
3	46.81	12.77	3	4	1.55	2.55	29.39	7.94	25.06	0.56	64.73	13.59
Barabási-Albert ($n = 50, \gamma = 3$): $\|E\| = 141$, max clique size = 4												
1	41.86	6.12	3	4	1.63	2.92	11.14	2.11	8.33	0.00	43.35	4.07
2	58.37	6.25	2	4	1.42	2.58	16.75	3.30	14.94	0.00	59.17	8.14
3	65.96	14.89	2	3	1.34	2.23	20.03	5.85	19.74	0.09	63.92	12.41
4	63.04	50.12	2	3	1.37	1.52	24.35	24.02	21.60	7.63	67.35	41.59
Barabási-Albert ($n = 75, \gamma = 2$): $\|E\| = 146$, max clique size = 3												
1	27.03	2.70	3	5	1.84	3.74	17.95	0.33	13.07	0.00	50.95	2.67
2	45.21	2.74	3	5	1.62	3.44	28.35	1.53	24.92	0.00	70.59	5.30
3	52.17	12.59	3	4	1.49	2.40	31.43	6.62	29.89	0.98	75.60	12.74
Barabási-Albert ($n = 75, \gamma = 3$): $\|E\| = 216$, max clique size = 4												
1	35.13	4.05	3	5	1.72	3.28	11.14	1.16	5.94	0.00	39.14	2.69
2	53.42	5.48	2	4	1.48	2.73	16.75	2.36	13.60	0.00	58.88	5.40
3	61.11	8.33	2	4	1.40	2.47	19.74	3.47	17.39	0.08	62.99	8.63
4	59.15	43.66	2	3	1.41	1.58	21.60	17.05	15.25	4.81	58.13	33.62

CHAPTER 5
THE INFLUENTIAL CLIQUE PROBLEM

5.1 Preliminaries

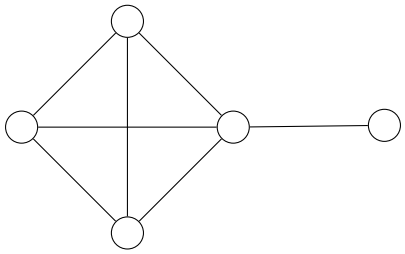
Let $G(V, E)$ be a simple, undirected graph on $|V| = n$ vertices and $|E| = m$ edges. A clique $C \subseteq V$ is a well-known graph structure, where any two nodes $i, j \in C$ are connected by an edge. The problem of detecting the maximum clique of a graph is an infamous $\mathcal{NP}$-hard problem that has attracted significant scientific and research interest over the years.

Further, denote the open neighborhood of a node $i \in V$ as $N(i) = \{j : (i, j) \in E\}$. The closed neighborhood of a node can be similarly defined as $N[i] = N(i) \cup \{i\}$. We can extend these definitions to a set of vertices $S \subseteq V$ as $N(S) = \{j : (i, j) \in E, \text{for some } i \in S, j \notin S\}$ and $N[S] = N(S) \cup S$. Last, we will be using $\bar{S}$ to signal the complement of a set of nodes S, i.e. $\bar{S} = V \setminus S$.

We define the problem of detecting a clique of maximum size, such that it is influential. A (k, l) influential clique is considered to be a well-connected clique inside a graph with at least k neighbors, and at least l nodes within the clique that are connected to nodes outside the clique. Informally, In our framework, this implies that a clique C can reach multiple nodes within one hop; and a series of nodes within the clique have at least one outgoing edge.

Note that, despite the similarities, this problem is inherently different than the dominating clique problem [35]. As an example, we refer the reader to Figure 5-1. In the figure, it is easy to see that while the graph has a dominating clique, that same clique would not be considered well–connected, since there is only one outgoing edge to one outside node. Further, it is easy to see that not every graph contains a dominating clique. On the contrary, a graph with a diameter bigger than 3 does not have a dominating clique, and is guaranteed to have one only when the graph at hand is P_5- and C_5-free [35].

Figure 5-1. Difference between a (k, l)-influential clique and a dominating clique.



5.2 Complexity

In the remainder of this chapter, we will be referring to standard graph theory notation. A graph will be represented as $G(\mathcal{V}, \mathcal{E})$, and when two nodes $i, j \in \mathcal{V}$ that are not adjacent, are connected with an edge in the complement graph, hence $(i, j) \in \bar{\mathcal{E}}$.

We define the problem $(\mathbf{k}, \mathbf{l})$-INFLUENTIAL CLIQUE as follows.

Definition 6. $(\mathbf{k}, \mathbf{l})$-INFLUENTIAL CLIQUE: Given a graph $G(V, E)$ and two integer numbers $k > 0$ and $l > 0$, is there a clique $C \subseteq V$ such that $|N(C)| \geq k$ and $|N(\bar{C})| \geq l$?

5.2.1 (k, l)-Influential Clique

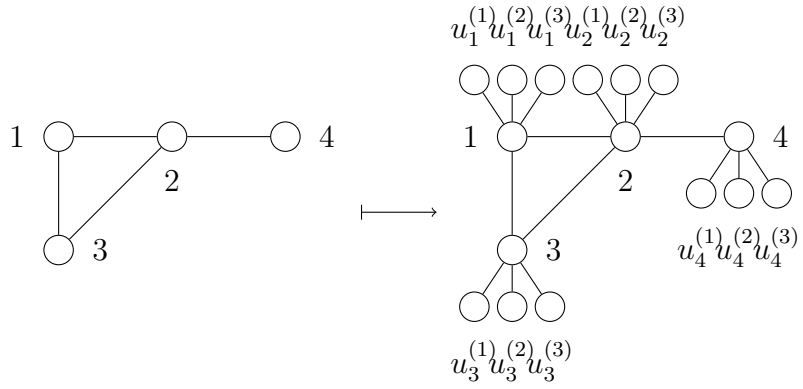
We will show the $\mathcal{NP}$–completeness of the problem using a reduction from the well–studied CLIQUE problem.

Theorem 5.1. $(\mathbf{k}, \mathbf{l})$-INFLUENTIAL CLIQUE is $\mathcal{NP}$–complete for any $k, l > 0$.

Proof. First of all, observe that the problem is in NP. Given a set of vertices $C \subseteq V$, it can be verified in polynomial time whether they form a clique, and if both $|N(C)| \geq k$ and $|N(\bar{C})| \geq l$.

Now, given a graph $G(\mathcal{V}, \mathcal{E})$, the following graph can be constructed in polynomial time. First, for every node $i \in \mathcal{V}$ introduce a vertex set $\hat{V}_i = \{u_i^{(j)}, j = 1, \ldots, k\}$. Overall, this action will add $n \cdot k$ new nodes in the graph. Then, we connect each node $i \in \mathcal{V}$ to every node that belongs to $\hat{V}_i$. Formally, we introduce the edge set $\hat{E}_i = \{(i, u_i^{(j)}, j =$

Figure 5-2. The gadget of the reduction for $k = 3$.



$1, \ldots, k\}$, for every $i \in \mathcal{V}$. Hence, we can now construct a new graph $\hat{G}(\hat{V}, \hat{E})$, where

$$\hat{V} = V \cup \{\bigcup_{i=1}^{n} \hat{V}_i\},$$

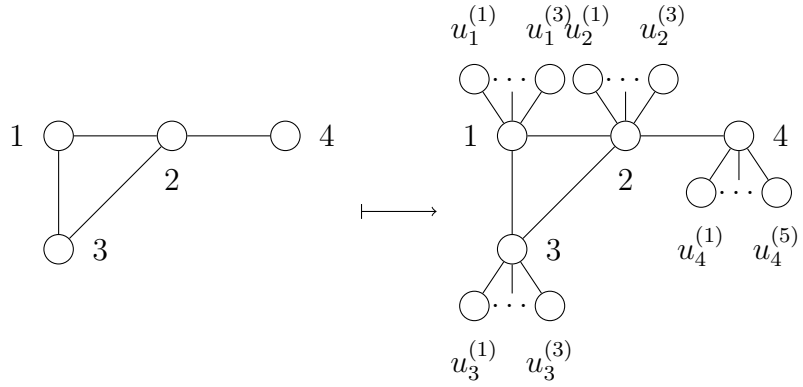
$$\hat{\mathcal{E}} = \mathcal{E} \cup \{\bigcup_{i=1}^{n} \hat{E}_i\}.$$

An example for $k = 3$ is shown in Figure 5-2.

The first part of the reduction is trivial. Assume that in graph G there exists a clique $C \subseteq \mathcal{V}$ such that $|C| \geq l$, then it is easy to see that C is also a clique in $\hat{G}$. Further, it is adjacent to at least k nodes. Taking any node $i \in C$, we immediately get $|\hat{V}_i| = k$ nodes that are adjacent to the clique. Last, the size of the clique C is at least l, and all nodes are connected to k nodes, which implies that $|N(\bar{C})| \geq l$.

For the converse, let us assume that there exists no clique in G with size greater than or equal to l. For a contradiction, suppose that there exists a clique C' in $\hat{G}$ such that $|N(C')| \geq k$ and $|N(\bar{C'})| \geq l$. Observe that we have $C' \not\subseteq \mathcal{V}$, otherwise C' would be a clique in G. Hence, there exists at least one node $u \in \hat{V} \setminus V$ that is part of the clique. However, note that by construction, this clique cannot have more than two nodes, since u will only be connected to one node $i \in \mathcal{V}$. Now, by assumption, clique C' has at least k nodes adjacent to it, since $N(C') \geq k$. There are $k - 1$ nodes remaining in the vertex set $\hat{V}_i$, since u belongs to the clique. Hence, there exists at least one node $u' \in \mathcal{V}$ that

Figure 5-3. Example of the reduction from G to $\hat{G}$ for $(k, 0)$–INFLUENTIAL CLIQUE.



is adjacent to i. Observe that i and u' form a clique C such that $|C| = |C'|$. The last observation contradicts the fact that there exists no clique $C \subseteq \mathcal{V}$ of size greater than or equal to l when there exists a clique $C' \subseteq \hat{\mathcal{V}}$ in $\hat{G}$.

□

5.2.2 $(k, 0)$-Influential Clique

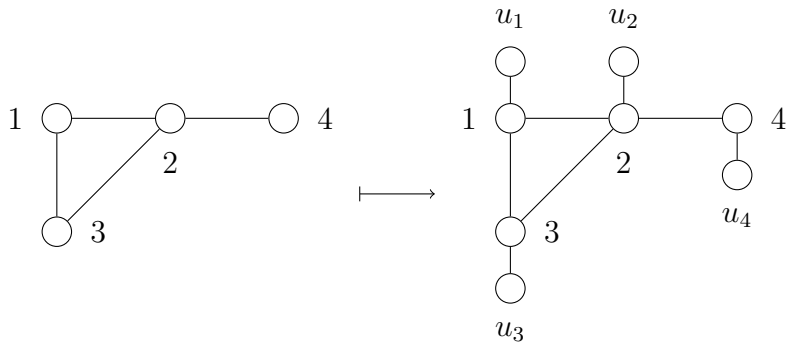
Theorem 5.2. $(k, 0)$-INFLUENTIAL CLIQUE is NP–complete, for all $k > 0$.

Proof. Once more, the problem is clearly in $\mathcal{NP}$, since it is trivial to check whether a set of vertices $C \subseteq V$ forms a clique and is connected to at least k nodes outside the clique.

We will show that the problem is NP–complete, reducing the CLIQUE problem to it. Let $< G, k >$, where $G(V, E)$ is a simple, non–empty graph, be an instance of CLIQUE. For every node $i \in V$, create $|V| + 1 = n + 1$ nodes $u_i^{(j)}, j = 1, \ldots, n + 1$ and connect them to i. That creates the graph $\hat{G}$. Now, we claim that graph G has a clique of size at least k iff $\hat{G}$ has a clique $\hat{C}$ such that $|N(C')| \geq k(n + 1)$, where $n = |V|$. An example of the construction is shown in Figure 5-3. Observe that G has $|V| = n = 4$ nodes, hence we create 5 new nodes in $\hat{G}$ for each node $i \in \mathcal{V}$.

First, assume that G has a clique $C \subseteq V$ of size at least k. Then, by construction, the same clique C covers at least $k(n + 1)$ nodes in $\hat{G}$, since all nodes belonging to the clique are adjacent to $n + 1$ nodes $u_i^{(j)}, j = 1, \ldots, n + 1$ outside the clique each. For the converse,

Figure 5-4. Example of the reduction from G to $\hat{G}$ for a graph with 4 nodes.



assume that G has no clique of size at least k, and, for a contradiction, there exists some clique C' in $\hat{G}$ such that $|N(C')| \geq k(n+1)$. Observe that there are two cases: i) $C' \subseteq V$, but $|C'| < k$, ii) C' contains some nodes $u_i^{(j)} \notin V$.

For the first case, since $|N(C')| \geq k(n+1)$ and $|C'| < k$, this implies that $|N(C') \cap V| > (k - |C'|)(n+1) \geq n+1$, which is a contradiction since $|N(C') \cap V| \leq |V| = n$. For the second case, observe that each of the constructed nodes $u_i^{(j)}$ are only connected to one node i. Hence, a clique consisting of these nodes would have a size of (at most) 2, and $|N(C')| = n$. Such a clique would be a solution only for $k = 0$, contradicting the definition of the problem that requires $k > 0$. □

5.2.3 $(0, l)$-Influential Clique

Theorem 5.3. $(0, l)$–INFLUENTIAL CLIQUE is NP–complete, for all $l > 0$.

Proof. Again, the problem is in $\mathcal{NP}$, since verifying that C forms a clique such that $|N(\bar{C})| \geq l$ can be performed in polynomial time.

Following a similar construction as before, given an instance of CLIQUE $< G, l >$, add one node u_i for every node $i \in V$, and connect (i, u_i) with an edge. Now, assume there exists a clique $C \subseteq V$ such that $|C| \geq l$, then it follows immediately that for the same clique in the constructed graph, we have $|N(C)| \geq l$. For the converse, assume for a contradiction that there is no clique of size at least l in the original graph, yet there exists a clique such that $|N(\bar{C}')| \geq l$. It is clear that $|C'| \geq |N(\bar{C}')| \geq l$. This implies that

C' is not a proper subset of V, otherwise C' would form a clique in the original graph, contradicting our assumption. Hence, there exists exactly one node u_i in C'. That would make (by construction) $C' = \{i, u_i\}$, for some $i \in V$. Observe, that if this is the case, then l must have been equal to 1. However, the graph is assumed to be nonempty, which implies that there definitely exists a clique of size 1 in G, finishing the proof. □

5.2.4 Inapproximability

Proposition 5.1. $(0, 0)$–INFLUENTIAL CLIQUE is equivalent to CLIQUE, i.e. finding a $(0, 0)$–INFLUENTIAL CLIQUE of size at least c is equivalent to finding a CLIQUE of size at least c, and is therefore NP–complete. On the other hand, finding a feasible solution is as easy as finding a clique in a graph.

Proposition 5.2. The maximum and minimum (k, l)–INFLUENTIAL CLIQUE problems and their special cases for $k = 0, l > 0$ or $l = 0, k > 0$ are inapproximable, under the assumption that $P \neq NP$.

Proof. It follows immediately from the $\mathcal{NP}$–completeness of finding a feasible solution to the problems. Assume there exists an algoritheorem $\mathcal{A}$ that returns any approximate solution to the problem. Hence, $\mathcal{A}$ also returns a feasible solution. Since detecting feasible solutions is $\mathcal{NP}$–complete, that would imply $P = NP$. □

Since the feasibility problems are $\mathcal{NP}$-complete, both minimization and maximization versions of the problems are $\mathcal{NP}$-hard.

5.2.5 A different complexity approach

Consider again the $(k, 0)$-INFLUENTIAL CLIQUE problem. We can see two different optimization versions of the problem that would be of interest:

- What is the biggest size clique C such that $|N(C)| \geq k$?
- What is the clique C of a specific size c with the biggest open neighborhood $|N(C)|$?

Observe that both problems described above have the same decision version. In the first problem, we aim to maximize the size of the clique in order to achieve a certain

influence requirement, whereas in the second one the influence is maximized subject to the restriction of the clique size being at least n. We will prove NP-completeness of the problem using the well-known MAXSAT problem.

Definition 7. Given a set of literals $x_i, i = 1, \ldots, n$ and their negations $\neg x_i, i = 1, \ldots, n$, and a set of clauses $C_j, j = 1, \ldots, m$, is there a valid assignment of literals such that at least k clauses are satisfied?

Theorem 5.4. MAXSAT is NP-complete [56].

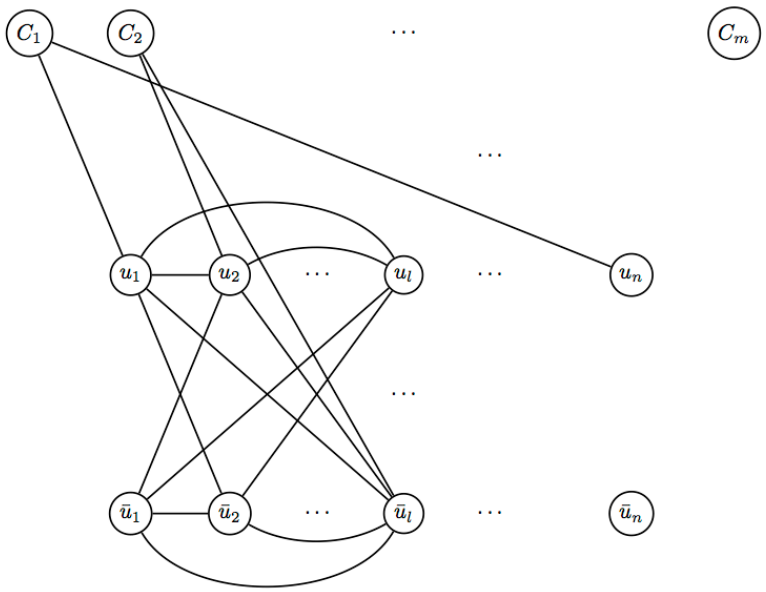
Theorem 5.5. (k, l)-INFLUENTIAL CLIQUE is NP-complete.

Proof. First of all, it is easy to see that our problem is in NP. Given a set of nodes C, it can be shown in polynomial time whether it is a valid solution, since we need to show that it is a clique of size n and has at least k other nodes that it is incident to.

Now, let us consider an instance of MAXSAT. For each of the literals x_i (and their negations, $\neg x_i$), create a node (u_i and $\bar{u}_i$ accordingly). Now, connect all nodes with every other node, but the node corresponding to the negation of their literal. Hence, connect each pair of (u_i, u_j), $i \neq j$, and $(u_i, \bar{u}_j)$, $i \neq j$. This will result in the creation of an n-partite subgraph, with two elements per partition. Now, create a node C_j for each of the clauses (m nodes). These nodes will be connected to the nodes of the literals that each of them consists of. As an example, if a clause contains x_1, x_3, and $\neg x_5$, the node corresponding to the clause will be connected to u_1, u_3, and $\bar{u}_5$. The constructed graph, starting from a MAXSAT instance, is shown in Figure 5-5. An example for the $MAXSAT$ clauses $C_1 = x_1 \vee x_2, C_2 = x_1 \vee \neg x_2, C_3 = \neg x_1 \vee x_2, C_4 = \neg x_1 \vee \neg x_2$ is shown in Figure 5-6. This instance can satisfy at most 3 clauses, and hence there is a clique that can influence 3 other nodes (but no more).

Now, we consider the case where a solution exists for the MAXSAT problem. In that case, we can have a set of n literals that satisfy at least k of the clauses. From the construction of the graph, selecting the nodes that correspond to these literals will yield

Figure 5-5. The gadget for the reduction from an instance of MAXSAT.

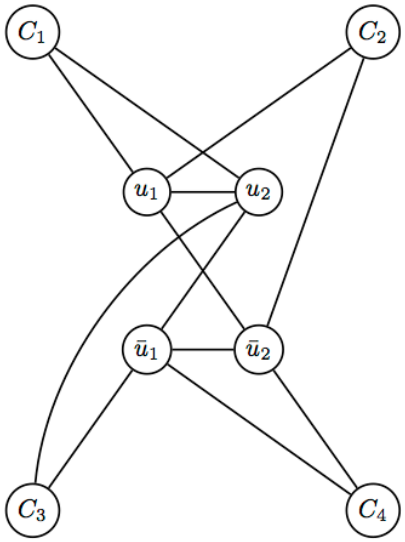


a clique of size n. Now, it is easy to check that the clique will be connected to k nodes (clauses) and n nodes (other literals), hence overall $n + k$ nodes. This implies that a solution to the MAXSAT problem with n literals and at least k clauses, yields a solution to our problem with size of clique n and coverage $l = n + k$.

Let us check the case where there is no solution to the MAXSAT problem. For a contradiction, assume that our problem yields a clique of size n such that the nodes in the clique are incident to $n + k$ other nodes. We distinguish two cases:

- the clique consists of only literal nodes (u, and v);
- the clique consists of literal and clause nodes (u, v, and C).

Figure 5-6. The constructed graph from the instance of MAXSAT with 2 literals (x_1 and $C_2 = x_1 \vee \neg x_2, C_3 = \neg x_1 \vee x_2, C_4 = \neg x_1 \vee \neg x_2$.



The first case can be dismissed, because by construction, if such a clique exists, then the original MAXSAT would have at least k satisfied clauses.

Let us concentrate on the second one. Observe, that there cannot exist more than one clause-node C in any valid clique. By construction, clause-nodes are not connected to each other, hence a set of nodes containing more than one clause-nodes would violate the condition of being a clique. Hence, there is exactly one clause-node in the clique. Further observe that each clause can have at most n literals, and cannot have a literal and its negation. Therefore, the clause-node can be connected at most to one other node. The rest of the nodes in the clique definitely cover the $n - 1$ remaining literal-nodes in the graph.

That leaves us with the rest of the nodes in the clique (the literal nodes) covering at least $k - 1$ clause-nodes. Now, observe that it would be a valid clique of size n, if we were to drop the clause-node currently in the clique, and add the literal-node (either the actual literal or its negation) in the clique. That way we would have a clique that covers at least $k - 1$ clause-nodes from the initially selected literal-nodes and at least 1 clause node from the last literal node. Adding to that the n literal nodes that are always covered because

of the construction, and we have a valid clique of size n that covers $n + k$ other nodes. However, that implies that our original MAXSAT problem also has a solution. Simply picking the literal-nodes that are in the clique would yield a valid assignment that satisfies at least k clauses.

□

We will not be tackling these problems herein, nonetheless we found the reduction to be of interest to the prospective reader.

5.3 Solution method

Let us define three sets of decision and auxiliary variables, as follows:

$$x_i = \begin{cases} 1, & \text{if } i \in V \text{ is selected to be in the clique} \\ 0, & \text{otherwise} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{if } i \in V \text{ is connected to at least one node inside the clique} \\ 0, & \text{otherwise} \end{cases}$$

$$z_i = \begin{cases} 1, & \text{if } i \in V \text{ is in the clique and is connected to at least one node outside the clique} \\ 0, & \text{otherwise} \end{cases}$$

Also, let k be the required coverage (influence) we aim to have, and l be the number of outgoing nodes within the clique.

5.3.1 Formulation

The problem of finding the maximum/minimum (k, l)–INFLUENTIAL CLIQUE can be easily formulated as shown in Formulation 5–1 throughout 5–2.

$$\max / \min \qquad f(x) = \sum_{i=1}^{n} x_i \tag{5–1}$$

$$s.t. \qquad x \in X_G^{k,l}, \tag{5–2}$$

where

$$
\begin{aligned}
X_G^{k,l} = \{x \in \{0,1\}^n : \quad & x_i + x_j \leq 1, (i,j) \in \bar{E} \\
& x_i + y_i \leq 1, i \in V \\
& y_i \leq \sum_{j:(i,j)\in E} x_j, i \in V \\
& \sum_{i=1}^{n} y_i \geq k \\
& z_i \leq x_i, i \in V \\
& z_i \leq \sum_{j:(i,j)\in E} (1 - x_j), i \in V \\
& \sum_{i=1}^{n} z_i \geq l \\
& y \in \{0,1\}^n \\
& z \in \{0,1\}^n\}.
\end{aligned}
$$

Let set $\mathcal{X}$ denote the relaxed X set, where variables y and z are allowed to take continuous value in $[0,1]$. The problem can be written as $\mathcal{R} : \{\min / \max f(x) : x \in \mathcal{X}\}$ Then, the following lemma is true.

Proposition 5.3. If $\mathcal{R}$ is feasible, then there always exists an optimal solution of $\mathcal{R}$ where all y and z variables take binary values.

Proof. Assume that $\mathcal{R}$ has an optimal solution (x^*, y^*, z^*), where some y_i and z_j are in $(0,1)$. Since $x \in \mathcal{X}$, we can deduce that $x_i = 0$, and $x_j = 1$. We also know that $\sum_{i=1}^{n} y_i^* \geq k$, and $\sum_{i=1}^{n} z_i^* \geq l$. Now, consider the solution $(x^*, \hat{y}, \hat{z})$, where

$$
\hat{y} = \begin{cases} 0, & \text{if } y_i^* = 0 \\ 1, & \text{if } y_i^* > 0 \end{cases}
$$

and

$$
\hat{z} = \begin{cases} 0, & \text{if } z_i^* = 0 \\ 1, & \text{if } z_i^* > 0 \end{cases}.
$$

Note that $(x^*, \hat{y}, \hat{z})$ is an alternate optimal solution, because $\sum_{i=1}^{n} \hat{y}_i > \sum_{i=1}^{n} y_i^* \geq k$, and $\sum_{i=1}^{n} \hat{z}_i > \sum_{i=1}^{n} z_i^* \geq l$. Hence, when y and z are not restricted to be integral, but are allowed to take continuous values, we can construct an optimal solution where y and z are binary.

□

5.3.2 Bounds

First of all, observe that finding a feasible solution in an instance $< G, k, l >$ is $\mathcal{NP}$–complete (cf. 5.2.1, 5.2.2, 5.2.3), hence we can only hope to find upper/lower bounds for the maximization/minimization problems respectively.

Assume an iterative method where S is the set of nodes that the clique under consideration consists of. Let L be the candidate list, of all nodes that can be added to S such that S remains a clique. Let $g(S, u)$ be a function defined as $g(S, u) = |N(S \cup \{u\})| - |N(S)|$. Then, the following is true.

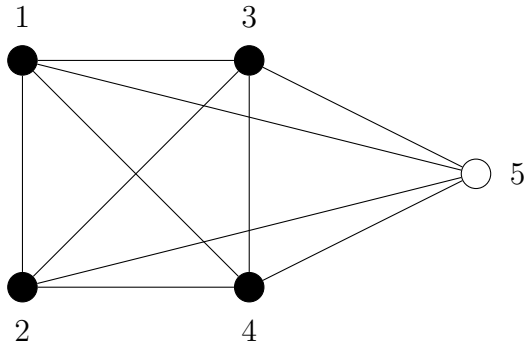
Proposition 5.4. $g(S, u) \geq -1, \forall u \in V \setminus S$.

It is easy to verify that a node does not decrease the size of the open neighborhood of S when added to it, iff $N(S) \cap N(u) \neq \emptyset$. However, in the case when $N(S) \cap N(u) = \emptyset$, then adding node u to S will decrease the size of the open neighborhood by exactly 1.

Now, let us consider the case for $N(\bar{C})$, for a clique $C \subseteq V$. First of all, it is trivial to see that in the case of (k, l)-INFLUENTIAL CLIQUE where $l > 0$, the size of the clique that we end up in any feasible solution should be at least equal to l, i.e. $|C| \geq l$, for any (k, l)–INFLUENTIAL CLIQUE.

Further, consider a node $u \notin C$, and let $S = C \cup \{u\}$. Then, observe that the size of $N(\bar{S})$ can decrease by at most $|C|$ compared to $|N(\bar{C})|$. As an example of this worst–case scenario, consider the graph in Figure 5-7. In this case, $C = \{1, 2, 3, 4\}$ and $|N(\bar{C})| = 4$. When $S = C \cup \{5\}$, note that $N(\bar{S}) = \emptyset$. Also observe that adding a node to S can at most increase the number of nodes that have connections outside the clique by 1. Let $h(S, u)$ be a function defined as $h(S, u) = |N(\overline{S \cup \{u\}})| - |N(\bar{S})|$.

Figure 5-7. An example of how the decrease of the size of $N(\bar{C})$ can be as big as $|C|$.



Proposition 5.5. $-|S| \leq h(S,u) \leq 1$, for all $u \in V \setminus S$.

In the following propositions, we exploit these observations. Assume we have a clique S and a candidate list L of all the nodes $v \in V \setminus S$ such that $S \cup \{v\}$ remains a clique. Further, we can sort the nodes belonging to L based on their g and h value. We will refer to these ordered lists as $\mathcal{L}_g$ and $\mathcal{L}_h$ respectively. An example is given in Figure .

Proposition 5.6. (a) Let $\mathcal{L}_g = \{v_1, \ldots, v_{|L|}\}$ be the ordered candidate list L, such that $g(S,v_i) \geq g(S,v_j)$, for any $v_i, v_j \in \mathcal{L}$, $i \leq j$, and let κ be the first index for which $g(S,v_\kappa) = 0$, or $\kappa = |L|+1$ otherwise. Then, $|N(S)| + \sum_{i=1}^{\kappa-1} g(S,v_i)$ is an upper bound on the size of the open neighborhood for S.

(b) Let $\mathcal{L}_h = \{v_1, \ldots, v_{|L|}\}$ be the ordered candidate list L, such that $h(S,v_i) \geq h(S,v_j)$, for any $v_i, v_j \in \mathcal{L}$, $i \leq j$, and let κ be the first index for which $h(S,v_\kappa) = 0$, or $\kappa = |L|+1$ otherwise. Then, $|N(\bar{S})| + \sum_{i=1}^{\kappa-1} h(S,v_i)$ is an upper bound on the size of the open neighborhood for $\bar{S}$.

Proof. (a) It follows immediately from the calculation. Each of the nodes $u_i \in L$ will increase the number of adjacent nodes by at most $g(S,u_i)$. In fact, for two sets $S \subseteq S'$, and a node $u \in V \setminus S'$ we have that $g(S',u) \leq g(S,u)$. If we add only the nodes that have a positive influence on the number of adjacent nodes we have a (loose) upper bound on the size of the open neighborhood that clique S can reach.

(b) Following a similar logic, each of the nodes $u_i \in L$ will increase the size of $N(\bar{S})$ by $h(S, u_i)$. Once more, like in (a), for two sets $S \subseteq S'$, and a node $u \in V \setminus S'$ we have that $h(S', u) \leq h(S, u)$. Hence, the maximum size of the open neighborhood of $\bar{S}$ that we can reach is $|N(\bar{S})| + \sum_{i=1}^{\kappa-1} h(S, v_i)$, since we are only adding the nodes that would increase the size of the open neighborhood.

□

Proposition 5.7. (a) If the upper bound on $N(S)$ is less than k, then there is no feasible solution containing S. Otherwise, let κ' be the first index for which $|N(S)|+\sum_{i=1}^{\kappa'} g(S, v_i) < k$ and $g(S, v'_\kappa) < 0$, or $\kappa' = |L| + 1$ otherwise, when $\mathcal{L}_g = \{v_1, v_2, \ldots, v_{|L|}\}$. Then, $|S \cup (\bigcup_{i=1}^{\kappa'-1} \{v_i\})|$ is an upper bound on the size of the (k, l)–INFLUENTIAL CLIQUE.

(b) If the upper bound on $N(\bar{S})$ is less than l, then there is no feasible solution containing S. Otherwise, let κ' be the first index for which $|N(\bar{S})| + \sum_{i=1}^{\kappa'} h(S, v_i) < l$ and $h(S, v'_\kappa) < 0$, or $\kappa' = |L| + 1$ otherwise, when $\mathcal{L}_h = \{v_1, v_2, \ldots, v_{|L|}\}$. Then, $|S \cup (\bigcup_{i=1}^{\kappa'-1} \{v_i\})|$ is an upper bound on the size of the (k, l)-INFLUENTIAL CLIQUE.

Proof. (a) It is easy to see that this is a valid upper bound on the size. The insight behind it has to do with the fact that even when adding new nodes to the clique S decreases the size of its open neighborhood by one, we can still go ahead and add them so long as the size does not drop below the k requirement.

(b) Similarly to before, we can proceed to add nodes in the clique, even when they decrease the size of the open neighborhood of $\bar{S}$, so long as it does not drop below l. □

Proposition 5.8. (a) Let $\mathcal{L}_h = \{v_1, \ldots, v_{|L|}\}$ be the ordered candidate list, such that $h(S, v_i) \geq h(S, v_j)$, for any $v_i, v_j \in \mathcal{L}$, $i \leq j$, and let κ be the first index for which $h(S, v_\kappa) < 0$, or $\kappa = |L| + 1$ otherwise. Then, $|N(\bar{S})| + \sum_{i=1}^{\kappa-1} g(S, v_i)$ is an upper bound on the size of the open neighborhood for $\bar{S}$.

(b) If the upper bound for the size of the open neighborhood is strictly less than k, then there can not exist a feasible solution containing S. Otherwise, let κ be the first

index for which $|N(S)| + \sum_{i=1}^{\kappa} g(S, v_i) < k$ and $g(S, v_\kappa) < 0$. Then, $|S \cup (\bigcup_{i=1}^{\kappa-1} \{v_i\})|$ is an upper bound on the size of the (k, l)-INFLUENTIAL CLIQUE.

From the above, we can see that for a given set S, if the upper bound of the open neighborhood as in Lemma 4 is strictly less than k, then the set S cannot produce a feasible solution to the problem.

Further, an obvious upper bound to the size of the optimal solution for the maximization version would be $|S| + |L|$. We can make this bound a little tighter as follows. Let $\mathcal{L}$ be the sorted candidate list again, and let h' be the index at which $|N(S)| + \sum_{i=1}^{h'} g(S, u_i) = k$, if it happens, or $h' = |L|$ otherwise. Then, we can have the following proposition for the upper bound on the size of the clique.

Proposition 5.9. $|S \cup (\bigcup_{i=1}^{h'} \{v_i\})|$ is an upper bound on the size of the maximum influential clique.

It is easy to see that this is a valid upper bound on the size. The insight behind it has to do with the fact that even when adding new nodes to the clique decreases the size of its open neighborhood by one, we can still go ahead and add them so long as the size does not drop below the k requirement.

5.3.3 Combinatorial Branch-and-Bound

First of all, let us make an important observation, that will lead us to a strong (as shown in the computational results) pre–processing scheme. Based on the notes above, given a requirement for l, we need to find a clique of at least size l. Hence, any node with a degree of $l - 1$ or less can never be part of a feasible clique. We can enhance on that bound a little more. Observe that each of the l nodes within a clique need to be connected to at least one node outside the clique. Hence, overall we have the following valid inequality.

Proposition 5.10. For all feasible solutions to the (k, l)–INFLUENTIAL CLIQUE, we have that $x_u = 0, \forall u \in V : deg(u) < l$.

Proof. In any feasible (k, l)–INFLUENTIAL CLIQUE C, there exist at least l nodes, each with a connection to some node $u \in V \setminus C$. These nodes have a degree of at least l. The remaining $|C| - l$ nodes (if any) have to be connected to at least $|C| - 1$ nodes each. Now observe that either $|C| = l$ or $|C| > l$. In the first case, all nodes have a degree of at least l, whereas in the latter all nodes have a degree of at least $|C| - 1 \geq l$.

□

Proposition 5.10 can also prove infeasibility for a problem. Assume there exist no more than $l - 1$ nodes that have a degree of at least l. This implies that there cannot be a clique of at least size l such that at least l nodes within the clique are adjacent to a node outside the clique. The procedure for setting some of the variables that cannot belong to the clique (preprocessing) is described in Algorithm 8.

Algorithm 8 Preprocessing(V)

```
S ← V
for i ∈ S do
    if deg(i) ≤ l then
        S ← S/{i}
    end if
end for
for i ∈ S do
    if deg(i) ≤ l − 1 then
        S ← S/{i}
    end if
end for
return S
```

For our branching scheme, we can either branch on the node that gives us the maximum upper bound for the size of the clique, the k-coverage ($\mathcal{L}_g$ list from before), or the l-coverage ($\mathcal{L}_h$ list from before). The process of the branching is described as follows. We first select the node with the highest clique size potential. If the maximum clique that can be obtained is smaller than the current incumbent, we can prune this branch by optimality. Then, we establish that its upper bound on both $N(S)$ and $N(\bar{S})$ coverage satisfy the k and l requirements. If not, we say that the tree branch is pruned by

infeasibility. Let p_i denote the potential clique size of a node $i \in V$, whereas UB^k and UB^l are the upper bounds for the k- and l- coverage, as shown in the previous section.

First of all, we apply Algorithm 8 to the vertex set V, leaving us with a set of "active" nodes S to choose from. At each node of the branch-and-bound tree, we select the vertex i with the highest potential p_i. This creates two branches: one where the clique C becomes $C \cup \{i\}$ and another where C stays the same, and $x_i = 0$. At that point, we distinguish the following cases:

- $p_i \leq incumbent$: we can prune this branch by optimality;
- $UB_i^k < k$ or $UB^l < l$: we can prune this branch by feasibility;
- $N(C) \geq k, N(\bar{C}) \geq l, |C| \geq incumbent$: the new incumbent would now be $|C|$.

5.4 Another formulation

We define $\mathcal{I}_G$ as the set of all maximal independent sets of graph (exponential in size). Further, let $\mathcal{S}_k$ and $\mathcal{S}_l$ be defined as the set of all subsets of nodes $S \subseteq V$ such that $|N(S)| < k$ and $|N(\bar{S})| < l$, respectively.

$$\min / \max \quad \sum_{i=1}^{n} x_i \tag{5–3}$$

$$s.t. \quad \sum_{i \in I} x_i \leq 1, \qquad \forall I \subseteq \mathcal{I}_G \tag{5–4}$$

$$\sum_{i \in S} (1 - x_i) + \sum_{i \in V \setminus S} x_i \geq 1, \qquad \forall S \subseteq \mathcal{S}_k \tag{5–5}$$

$$\sum_{i \in S} (1 - x_i) + \sum_{i \in V \setminus S} x_i \geq 1, \qquad \forall S \subseteq \mathcal{S}_l \tag{5–6}$$

$$x_i \in \{0, 1\}, \qquad \forall i \in V. \tag{5–7}$$

Constraint 5–4 is the typically used maximal independent set constraint to model a clique. In that, we state that at most one of the nodes in every maximal independent set can be in a clique. Constraints 5–5 and 5–6 model the requirements for coverage,

according to k and l. The first one ensures that for every subset of nodes S in V that do not satisfy $|N(S)| \geq k$, we should either add at least one node to S, or remove at least one node from S. Similarly, the latter forces the same for every subset of nodes S such that $|N(\bar{S})| < l$. Last, Constraint 5–7 ensures the binary nature of the x variables.

The model shown in Formulation 5–3 throughout 5–7 has exponentially many constraints, hence a proper framework is needed to tackle the problem. We employ the typically used constraint generation approach, which is summed up to the following. We start with a set of constraints $\mathcal{C}$, which are a proper subset of the original constraint set of the problem. We can then proceed to solve the smaller problem, and then employ an oracle $\mathcal{O}$ to provide us with a violated constraint from the set of constraints that were originally present. On the other hand, if the separation oracle never provides us with a violated constraint, we can terminate with an optimal solution. In our work, that is depicted in the computational results, we employ the method of most violated constraint [58].

5.5 Computational Results

In Table 5-1, we present our numerical findings. First of all, it is easy to see that the combinatorial branch-and-bound works much better even without the preprocessing. However, it has to be noted that when we preprocess the nodes based on Algorithm 8 the speedup is much higher (up to 300%). Last, the constraint generation approach based on the formulation of the previous section is indeed faster than the original formulation, yet fails to beat the combinatorial branch and bound.

The benchmarks used include karate (a social network shown in [123]), dolphins (a social network from [88], polbooks (a collection of books on US politics [82]), adjnoun (a collection of common adjective and noun adjacencies in "David Copperfield" [92]), and two book graphs, huck and jean from [1].

Table 5-1. Computational results for different values of k and l.

Graph	(k, l)	Exact	B&B w. Pre.	B&B wo. Pre.	Constraint Generation
polbooks	(10,3)	1.48	0.98	1.21	1.20
polbooks	(10,4)	1.30	0.76	1.03	0.99
polbooks	(20,3)	1.39	0.83	1.19	1.23
polbooks	(20,4)	1.06	0.53	0.77	0.78
polbooks	(20,5)	1.57	1.02	1.12	1.15
karate	(10,3)	0.02	0.01	0.01	0.01
karate	(10,4)	0.15	0.10	0.11	0.11
karate	(15,3)	0.03	0.01	0.02	0.02
karate	(15,4)	0.04	0.03	0.04	0.03
karate	(20,3)	0.15	0.11	0.13	0.13
dolphins	(10,3)	0.28	0.15	0.20	0.19
dolphins	(10,4)	0.35	0.21	0.28	0.27
dolphins	(10,5)	0.07	0.04	0.05	0.06
dolphins	(15,3)	0.25	0.19	0.23	0.25
dolphins	(15,4)	0.20	0.13	0.18	0.14
dolphins	(15,5)	0.05	0.04	0.06	0.05
huck	(20,5)	0.13	0.07	0.10	0.11
huck	(20,6)	0.14	0.09	0.11	0.11
huck	(30,5)	0.21	0.17	0.19	0.19
huck	(30,6)	0.23	0.15	0.17	0.17
jean	(20,5)	0.12	0.09	0.12	0.10
jean	(20,6)	0.13	0.09	0.12	0.11
jean	(30,5)	0.13	0.10	0.12	0.12
jean	(30,6)	0.14	0.10	0.13	0.12
adjnoun	(15,4)	0.28	0.21	0.27	0.26
adjnoun	(15,5)	0.35	0.23	0.31	0.31
adjnoun	(20,4)	0.27	0.20	0.25	0.26
adjnoun	(20,5)	0.35	0.25	0.31	0.31
adjnoun	(30,4)	0.30	0.21	0.26	0.27
adjnoun	(30,5)	0.33	0.24	0.27	0.28

5.6 Future work

In this chapter, we introduced and tackled the novel problem of detecting (k, l)-influential cliques in graph. We proceeded to provide the complexity of the problem (along with several special cases), and formulated it mathematically. Then, we showed mathematical bounds on both the size of such cliques and their influence requirements that we employed in a combinatorial branch-and-bound scheme. The numerical results in well-known

benchmark networks reveal that our approach is a viable and efficient way to solving these problems.

CHAPTER 6
CONCLUSION

In this dissertation, we studied several problems that arise from the widespread use of sensors in the modern world. Such problems admit advanced operations research and optimization techniques to ensure the longevity and efficiency of sensor networks.

More specifically, in Chapter 2, we gave an annotated literature review of the use of sensors in modern transportation and logistics networks. This review can serve as a starting point for realizing how modern sensor systems work, and the problems that practitioners face today.

Then, we proceeded in Chapter 3 to state the well-known multi-sensor multi-target tracking problem, where a set of sensor measurements need to be clustered together to most accurately track a series of targets. We stated the problem in its graph theoretic terms, and proposed two novel decomposition schemes. These subproblems were shown to provide us with valid upper and lower bounds for the original problem. Further, we research exact and heuristic methods for solving the subproblems and recombining the solutions to obtain feasible (approximate) and exact solutions to the master problem. These approaches, along with a hybrid method that combines elements from both partitioning schemes, were shown to be very efficient for solving realistic, large-scale MAP instances.

In Chapter 4, we dealt with the very interesting problem of group centrality. More specifically, we extended the definitions of degree, closeness, and betweenness centrality to cohesive groups (cliques). We showed that the problems remain NP-hard when the group is restricted to form a clique, and proposed mathematical programing formulations for all three problems. Then, as far as betweenness centrality was concerned, we proposed novel interpretations, namely probabilistic and pessimistic, instead of the typically used optimistic version. Finally, our experiments depicted that it is not always that bigger

cliques provide the same level of coverage, closeness, and information tracking as smaller, yet "central" cliques.

Last, in Chapter 5, the problem of a (k, l)-Influential clique was introduced and studied. We showed that the problem in its native form is NP-hard, along with several special cases that stem from it. We also proceeded to show that the problem is inapproximable, under the long-standing assumption that $P \neq NP$. Then, we proposed a mathematical formulation, and enhanced it with a series of bounds. These bounds actually help propose a combinatorial branch-and-bound, that was shown to outperform CPLEX and Gurobi in the computational results.

REFERENCES

[1] "COLOR02/03/04: Graph Coloring and its Generalizations." http://mat.gsia.cmu.edu/COLOR03/, . Last accessed September 9, 2013.

[2] Abello, J., Pardalos, P. M., and Resende, M. G. C. "On Maximum Clique Problems In Very Large Graphs." In External Memory Algorithms. vol. 50 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 1999, 119–130.

[3] Aiex, R.M., Resende, M.G.C., Pardalos, PM, and Toraldo, G. "GRASP with path relinking for the three-index assignment problem." INFORMS Journal on Computing (2000).

[4] Al-Karaki, Jamal N and Kamal, Ahmed E. "Routing techniques in wireless sensor networks: a survey." Wireless Communications, IEEE 11 (2004).6: 6–28.

[5] Andrey, Philippe and Tarroux, Philippe. "Unsupervised segmentation of Markov random field modeled textured images using selectionist relaxation." Pattern Analysis and Machine Intelligence, IEEE Transactions on 20 (1998).3: 252–262.

[6] Andrijich, Sharlene M. and Caccetta, Louis. "Solving the multisensor data association problem." Nonlinear Analysis 47 (2001).8: 5525–5536.

[7] Bandelt, Hans-Jürgen, Crama, Yves, and Spieksma, Frits CR. "Approximation algorithms for multi-dimensional assignment problems with decomposable costs." Discrete Applied Mathematics 49 (1994).1: 25–50.

[8] Bar-Shalom, Yaakov. "Multitarget-multisensor tracking: advanced applications." Norwood, MA, Artech House, 1990, 391 p. 1 (1990).

[9] Bar-Shalom, Yaakov, Chummun, Muhammad R., Kirubarajan, Thiagalingam, and Pattipati, Krishna R. "Fast Data Association for Multisensor-Multitarget Tracking Using Clustering and Multidimensional Assignment Algorithms." IEEE Transactions on Aerospace and Electronic Systems 37 (2001).3: 898–913.

[10] Baskar, Lakshmi Dhevi, De Schutter, Bart, and Hellendoorn, Hans. "Hierarchical traffic control and management with intelligent vehicles." Intelligent Vehicles Symposium, 2007 IEEE. IEEE, 2007, 834–839.

[11] ———. "Optimal routing for intelligent vehicle highway systems using mixed integer linear programming." Proceedings of the 12th IFAC Symposium on Transportation Systems. 2009, 569–575.

[12] Baskar, Lakshmi Dhevi, De Schutter, Bart, Hellendoorn, J, and Tarau, A. "Traffic management for intelligent vehicle highway systems using model-based predictive control." Proceedings of the 88th Annual Meeting of the Transportation Research Board. 2009.

[13] Bassett, Danielle S, Owens, Eli T, Daniels, Karen E, and Porter, Mason A. "Influence of network topology on sound propagation in granular materials." Physical Review E 86 (2012).4: 041306.

[14] Bavelas, Alex. "A mathematical model for group structures." Human organization 7 (1948).3: 16–30.

[15] ———. "Communication patterns in task-oriented groups." The Journal of the Acoustical Society of America 22 (1950).6: 725–730.

[16] Bemporad, Alberto and Morari, Manfred. "Control of systems integrating logic, dynamics, and constraints." Automatica 35 (1999).3: 407–427.

[17] Blackman, Samuel S. Multiple-target tracking with radar applications. Artech House, Inc., 1986.

[18] Boginski, V., Butenko, S., and Pardalos, P. "Mining market data: A network approach." Computers & Operations Research 33 (2006).11: 3171–3184.

[19] Boginski, Vladimir L, Commander, Clayton W, and Pardalos, Panos M. Sensors: theory, algorithms, and applications, vol. 61. Springer, 2012.

[20] Boldi, Paolo and Vigna, Sebastiano. "Axioms for centrality." arXiv preprint arXiv:1308.2140 (2013).

[21] Bomze, Immanuel M., Budinich, Marco, Pardalos, Panos M., and Pelillo, Marcello. "The Maximum Clique Problem." Handbook of Combinatorial Optimization. Kluwer Academic Publishers, 1999, 1–74.

[22] Borgatti, Stephen P and Everett, Martin G. "A graph-theoretic perspective on centrality." Social networks 28 (2006).4: 466–484.

[23] Braginsky, David and Estrin, Deborah. "Rumor routing algorthim for sensor networks." Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications. ACM, 2002, 22–31.

[24] Brandes, Ulrik. "A faster algorithm for betweenness centrality*." Journal of Mathematical Sociology 25 (2001).2: 163–177.

[25] Buchanan, Austin, Sung, Je Sang, Butenko, S, Boginski, V, and Pasiliao, E. "On connected dominating sets of restricted diameter." Tech. rep., Working paper, 2012.

[26] Bullo, Francesco, Cortés, Jorge, and Martınez, Sonia. "Distributed control of robotic networks." Applied Mathematics Series. Princeton University Press (2009).

[27] Burkard, Rainer Ernst and Çela, Eranda. "Quadratic and three-dimensional assignment problems: An annotated bibliography." Annotated Bibliographies in Combinatorial Optimization. eds. M. DellAmico, F. Maffioli, and S. Martello, chap. 21. John Wiley & Sons, 1997. 373–392.

[28] ———. "Linear assignment problems and extensions." Handbook of Combinatorial Optimization. eds. Dingzhu Du and Panos M. Pardalos, chap. 21. Kluwer Academic Publishers, 1999. 75–149.

[29] Butenko, Sergiy and Wilhelm, Wilbert E. "Clique-detection models in computational biochemistry and genomics." European Journal of Operational Research 173 (2006).1: 1–17.

[30] Çela, Eranda. "Assignment Problems." Handbook of Applied Optimization. eds. Panos M. Pardalos and Mauricio G. C. Resende, chap. 17.9. New York NY: Oxford University Press, 2002. 661–678.

[31] Chen, Lei, Wainwright, Martin J, Cetin, Mujdat, and Willsky, Alan S. "Multitarget-multisensor data association using the tree-reweighted max-product algorithm." AeroSense 2003. International Society for Optics and Photonics, 2003, 127–138.

[32] Chu, Maurice, Haussecker, Horst, and Zhao, Feng. "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks." International Journal of High Performance Computing Applications 16 (2002).3: 293–313.

[33] Coleri, Sinem, Cheung, Sing Yiu, and Varaiya, Pravin. "Sensor networks for monitoring traffic." Allerton conference on communication, control and computing. 2004, 32–40.

[34] Costa, L da F, Rodrigues, Francisco A, Travieso, Gonzalo, and Villas Boas, PR. "Characterization of complex networks: A survey of measurements." Advances in Physics 56 (2007).1: 167–242.

[35] Cozzens, Margaret B. and Kelleher, Laura L. "Dominating cliques in graphs." Discrete Mathematics 86 (1990).13: 101 – 116.

[36] Dangalchev, Chavdar. "Residual closeness in networks." Physica A: Statistical Mechanics and its Applications 365 (2006).2: 556–564.

[37] Davis, T.A. and Hu, Y. "The University of Florida Sparse Matrix Collection." ACM Transactions on Mathematical Software 38 (2011).1: 1–25.

[38] DIMACS. "10th DIMACS Implementation Challenge." Available at http://www.cc.gatech.edu/dimacs10/index.shtml, last accessed September, 2013, 2011.

[39] Dixon, Cory and Frew, Eric W. "Maintaining optimal communication chains in robotic sensor networks using mobility control." Mobile Networks and Applications 14 (2009).3: 281–291.

[40] Dolev, Shlomi, Elovici, Yuval, Puzis, Rami, and Zilberman, Polina. "Incremental deployment of network monitors based on group betweenness centrality." Information Processing Letters 109 (2009).20: 1172–1176.

[41] Ercsey-Ravasz, Mária, Lichtenwalter, Ryan N, Chawla, Nitesh V, and Toroczkai, Zoltán. "Range-limited centrality measures in complex networks." Physical Review E 85 (2012).6: 066103.

[42] Erdős, P. and Rényi, A. "On random graphs." Publicationes Mathematicae Debrecen 6 (1959): 290–297.

[43] Ergen, Sinem Coleri and Varaiya, Pravin. "Optimal placement of relay nodes for energy efficiency in sensor networks." Communications, 2006. ICC'06. IEEE International Conference on. vol. 8. IEEE, 2006, 3473–3479.

[44] Estrada, Ernesto. "Path Laplacian matrices: introduction and application to the analysis of consensus in networks." Linear Algebra and its Applications 436 (2012).9: 3373–3391.

[45] Everett, Martin G and Borgatti, Stephen P. "The centrality of groups and classes." The Journal of mathematical sociology 23 (1999).3: 181–201.

[46] ———. "Extending centrality." Models and methods in social network analysis 35 (2005).1: 57–76.

[47] Faruque, Jabed, Psounis, Konstantinos, and Helmy, Ahmed. "Analysis of gradient-based routing protocols in sensor networks." Distributed Computing in Sensor Systems. Springer, 2005. 258–275.

[48] Fasolo, Elena, Rossi, Michele, Widmer, Jorg, and Zorzi, Michele. "In-network aggregation techniques for wireless sensor networks: a survey." Wireless Communications, IEEE 14 (2007).2: 70–87.

[49] Fink, Martin and Spoerhase, Joachim. "Maximum betweenness centrality: approximability and tractable cases." WALCOM: Algorithms and Computation. Springer, 2011. 9–20.

[50] Floyd, Robert W. "Algorithm 97: shortest path." Communications of the ACM 5 (1962).6: 345.

[51] Freeman, Linton C. "A set of measures of centrality based on betweenness." Sociometry (1977): 35–41.

[52] ———. "Centrality in social networks conceptual clarification." Social Networks 1 (1979).3: 215 – 239.

URL http://www.sciencedirect.com/science/article/pii/0378873378900217

[53] Gaonkar, Shravan, Li, Jack, Choudhury, Romit Roy, Cox, Landon, and Schmidt, Al. "Micro-blog: sharing and querying content through mobile phones and social participation." Proceedings of the 6th international conference on Mobile systems, applications, and services. ACM, 2008, 174–186.

[54] Gardiner, Eleanor J, Artymiuk, Peter J, and Willett, Peter. "Clique-detection algorithms for matching three-dimensional molecular structures." Journal of Molecular Graphics and Modelling 15 (1997).4: 245–253.

[55] Gardner, Warren F and Lawton, Daryl T. "Interactive model-based vehicle tracking." (1995).

[56] Garey, M. and Johnson, D. Computers and Intractability: A Guide to the Theory of NP-completeness. Freeman and Co., New York, 1979.

[57] Geman, Stuart and Geman, Donald. "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images." Pattern Analysis and Machine Intelligence, IEEE Transactions on (1984).6: 721–741.

[58] Geoffrion, Arthur M. "Generalized benders decomposition." Journal of optimization theory and applications 10 (1972).4: 237–260.

[59] Gerkey, Brian P and Mataric, Maja J. "Sold!: Auction methods for multirobot coordination." Robotics and Automation, IEEE Transactions on 18 (2002).5: 758–768.

[60] Gilbert, Kenneth and Hofstra, R. "Multidimensional Assignment Models." Decision Sciences 19 (1988): 306–321.

[61] Goldengorin, Boris, Kalyagin, Valery A, and Pardalos, Panos M. "Models, Algorithms, and Technologies for Network Analysis." (2013).

[62] Grundel, D.A. and Pardalos, P.M. "Test problem generator for the multidimensional assignment problem." Computational Optimization and Applications 30 (2005).2: 133–146.

[63] Hagberg, Aric, Swart, Pieter, and S Chult, Daniel. "Exploring network structure, dynamics, and function using NetworkX." Tech. rep., Los Alamos National Laboratory (LANL), 2008.

[64] Hage, Per and Harary, Frank. "Eccentricity and centrality in networks." Social networks 17 (1995).1: 57–63.

[65] Hirsch, Michael J, Pardalos, Panos M, and Resende, Mauricio GC. "Sensor registration in a sensor network by continuous GRASP." Military Communications Conference, 2006. MILCOM 2006. IEEE. IEEE, 2006, 1–6.

[66] Holder, Lawrence B and Cook, Diane J. "Graph-Based Data Mining." Encyclopedia of data warehousing and mining 2 (2009): 943–949.

[67] Hummel, B. "Map matching for vehicle guidance." Dynamic and Mobile GIS: Investigating Space and Time (2006): 437–438.

[68] Inalhan, Gokhan, Stipanovic, Dusan M, and Tomlin, Claire J. "Decentralized optimization, with application to multiple aircraft coordination." Decision and Control, 2002, Proceedings of the 41st IEEE Conference on. vol. 1. IEEE, 2002, 1147–1155.

[69] Intanagonwiwat, Chalermek, Estrin, Deborah, Govindan, Ramesh, and Heidemann, John. "Impact of network density on data aggregation in wireless sensor networks." Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on. IEEE, 2002, 457–458.

[70] Intanagonwiwat, Chalermek, Govindan, Ramesh, and Estrin, Deborah. "Directed diffusion: a scalable and robust communication paradigm for sensor networks." Proceedings of the 6th annual international conference on Mobile computing and networking. ACM, 2000, 56–67.

[71] Jacob, Riko, Koschützki, Dirk, Lehmann, Katharina Anna, Peeters, Leon, and Tenfelde-Podehl, Dagmar. "Algorithms for centrality indices." Network Analysis. Springer, 2005. 62–82.

[72] Kamijo, Shunsuke, Matsushita, Yasuyuki, Ikeuchi, Katsushi, and Sakauchi, Masao. "Traffic monitoring and accident detection at intersections." Intelligent Transportation Systems, IEEE Transactions on 1 (2000).2: 108–118.

[73] Karp, Richard M. "On-line algorithms versus off-line algorithms: How much is it worth to know the future?" IFIP Congress (1). vol. 12. 1992, 416–429.

[74] Kass, Michael, Witkin, Andrew, and Terzopoulos, Demetri. "Snakes: Active contour models." International journal of computer vision 1 (1988).4: 321–331.

[75] Klavins, Eric. "Communication complexity of multi-robot systems." Algorithmic Foundations of Robotics V. Springer, 2004. 275–292.

[76] Kolaczyk, Eric D, Chua, David B, and Barthélemy, Marc. "Group betweenness and co-betweenness: Inter-related notions of coalition centrality." Social Networks 31 (2009).3: 190–203.

[77] Koschützki, Dirk, Lehmann, Katharina Anna, Peeters, Leon, Richter, Stefan, Tenfelde-Podehl, Dagmar, and Zlotowski, Oliver. "Centrality indices." Network analysis. Springer, 2005. 16–61.

[78] Kratsch, Dieter. "Finding dominating cliques efficiently, in strongly chordal graphs and undirected path graphs." Discrete Mathematics 86 (1990).13: 225 – 238.

[79] ———. "Algorithms." Domination in graphs: advanced topics. eds. Hedetniemi S. T. Haynes, T. W. and P. J. Slater. 1998.

[80] Kratsch, Dieter and Liedloff, Mathieu. "An Exact Algorithm for the Minimum Dominating Clique Problem." Parameterized and Exact Computation. eds. Hans L.

Bodlaender and Michael A. Langston, vol. 4169 of Lecture Notes in Computer Science. 2006. 130–141.

[81] Krebs, V. "Uncloaking Terrorist Networks." First Monday 7 (2002).4. Available at http://journals.uic.edu/ojs/index.php/fm/article/view/941, last accessed September 9, 2013.

[82] ———. "Dataset: PolBooks–Krebs." .

[83] Krumm, John, Letchner, Julie, and Horvitz, Eric. "Map matching with travel time constraints." SAE World Congress. 2007.

[84] Larsen, Morten. "Branch and bound solution of the multidimensional assignment problem formulation of data association." Optimization Methods and Software 27 (2012).6: 1101–1126.

[85] Leavitt, Harold J. "Some effects of certain communication patterns on group performance." The Journal of Abnormal and Social Psychology 46 (1951).1: 38.

[86] Li, Deying, Liu, Lin, and Yang, Huiqiang. "Minimum connected r-hop k-dominating set in wireless networks." Discrete Mathematics, Algorithms and Applications 1 (2009).01: 45–57.

[87] Luce, R Duncan and Perry, Albert D. "A method of matrix analysis of group structure." Psychometrika 14 (1949).2: 95–116.

[88] Lusseau, D., Schneider, K., Boisseau, O.J., Haase, P., Slooten, E., and Dawson, S.M. "The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations." Behavioral Ecology and Sociobiology 54 (2003).4: 396–405.

URL http://dx.doi.org/10.1007/s00265-003-0651-y

[89] MacDuffie, John Paul and Krafcik, John. "Integrating technology and human resources for high-performance manufacturing: Evidence from the international auto industry." Transforming organizations (1992): 209–226.

[90] Mimbela, Luz Elena Y and Klein, Lawrence A. "Summary of vehicle detection and surveillance technologies used in intelligent transportation systems." (2000).

[91] Nadeem, Tamer, Dashtinezhad, Sasan, Liao, Chunyuan, and Iftode, Liviu. "TrafficView: traffic data dissemination using car-to-car communication." ACM SIGMOBILE Mobile Computing and Communications Review 8 (2004).3: 6–19.

[92] Newman, Mark EJ. "Finding community structure in networks using the eigenvectors of matrices." Physical review E 74 (2006).3: 036104.

[93] Oliveira, C.A.S. and Pardalos, P.M. "Randomized parallel algorithms for the multidimensional assignment problem." Applied Numerical Mathematics 49 (2004).1: 117–133.

[94] Oliveira, Eduardo MR, Ramos, Heitor S, and Loureiro, Antonio Alfredo Ferreira. "Centrality-based routing for wireless sensor networks." Wireless Days (WD), 2010 IFIP. IEEE, 2010, 1–5.

[95] O'Toole, Randal. Gridlock: why we're stuck in traffic and what to do about it. Cato Institute, 2010.

[96] Pagani, Giuliano Andrea and Aiello, Marco. "The power grid as a complex network: a survey." Physica A: Statistical Mechanics and its Applications (2013).

[97] Pardalos, Panos M. and Pitsoulis, Leonidas S., eds. Nonlinear Assignment Problems: Algorithms and Applications, vol. 7 of Combinatorial Optimization. Dordrecht: Kluwer Academic Publishers, 2000.

[98] Pasiliao, Eduardo L. Algorithms for multidimensional assignment problems. Ph.D. thesis, University of Florida, 2003.

[99] ———. "Local Neighborhoods for the Multidimensional Assignment Problem." Dynamics of Information Systems (2010): 353–371.

[100] Pasiliao, Eduardo L., Pardalos, P.M., and Pitsoulis, L.S. "Branch and bound algorithms for the multidimensional assignment problem." Optimization Methods and Software 20 (2005).1: 127–143.

[101] Pattillo, Jeffrey, Youssef, Nataly, and Butenko, Sergiy. "On clique relaxation models in network analysis." European Journal of Operational Research 226 (2013).1: 9–18.

[102] Peterfreund, Natan. "Robust tracking of position and velocity with Kalman snakes." Pattern Analysis and Machine Intelligence, IEEE Transactions on 21 (1999).6: 564–569.

[103] Phillips, Don T and Garcia-Diaz, Alberto. Fundamentals of network analysis, vol. 198. Prentice-Hall Englewood Cliffs, NJ, 1981.

[104] Pierskalla, William P. "The multidimensional assignment problem." Operations Research 16 (1968).2: 422–431.

[105] Pitsoulis, Leonidas S. and Resende, Mauricio G. C. "Greedy Randomized Adaptive Search Procedures." Handbook of Applied Optimization. eds. Panos M. Pardalos and Mauricio G. C. Resende, chap. 3.6.5. New York NY: Oxford University Press, 2002. 168–183.

[106] Poore, Aubrey B. "Multidimensional assignment formulation of data association problems arising from multitarget and multisensor tracking." Computational Optimization and Applications 3 (1994): 27–57.

[107] ———. "Multidimensional assignment formulation of data association problems arising from multitarget and multisensor tracking." Computational Optimization and Applications 3 (1994).1: 27–57.

[108] Poore, Aubrey B. and III, Alexander J. Robertson. "A New Lagrangian relaxation based algorithm for a class of multidimensional assignment problems." Computational Optimization and Applications 8 (1997): 129–150.

[109] Puzis, Rami, Elovici, Yuval, and Dolev, Shlomi. "Finding the most prominent group in complex networks." AI communications 20 (2007).4: 287–296.

[110] Puzis, Rami, Yagil, Dana, Elovici, Yuval, and Braha, Dan. "Collaborative attack on Internet users' anonymity." Internet Research 19 (2009).1: 60–77.

[111] Qu, Zhihva. Cooperative control of dynamical systems. Springer, 2009.

[112] Reka, A. and Barabási, A-L. "Statistical mechanics of complex networks." Reviews of Modern Physics 74 (2002): 47–97.

URL http://arxiv.org/abs/cond-mat/0106096

[113] Resende, Mauricio GC and Pardalos, Panos M. Handbook of applied optimization. Oxford university press, 2002.

[114] Sabidussi, Gert. "The centrality index of a graph." Psychometrika 31 (1966).4: 581–603.

[115] Sadagopan, Narayanan, Krishnamachari, Bhaskar, and Helmy, Ahmed. "Active query forwarding in sensor networks." Ad Hoc Networks 3 (2005).1: 91–113.

[116] Sharma, Vikrant, Savchenko, Michael A, Frazzoli, Emilio, and Voulgaris, Petros G. "Time Complexity of Sensor-Based Vehicle Routing." Robotics: Science and Systems. Citeseer, 2005, 297–304.

[117] Spieksma, Frits C. R. "Multi index assignment problems: Complexity, approximation, applications." Nonlinear Assignment Problems: Algorithms and Applications. eds. Panos M. Pardalos and Leonidas S. Pitsoulis, vol. 7 of Combinatorial Optimization, chap. 1. Dordrecht: Kluwer Academic Publishers, 2000. 1–12.

[118] Thiagarajan, Arvind, Ravindranath, Lenin, LaCurts, Katrina, Madden, Samuel, Balakrishnan, Hari, Toledo, Sivan, and Eriksson, Jakob. "VTrack: accurate, energy-aware road traffic delay estimation using mobile phones." Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems. ACM, 2009, 85–98.

[119] Vivaldini, Kelen CT, Galdames, Jorge PM, Bueno, Thales S, Araújo, Roberto C, Sobral, Rafael M, Becker, Marcelo, and Caurin, Glauco AP. "Robotic forklifts for intelligent warehouses: Routing, path planning, and auto-localization." Industrial Technology (ICIT), 2010 IEEE International Conference on. IEEE, 2010, 1463–1468.

[120] Vogiatzis, Chrysafis. "Sensors in Transportation and Logistics Networks." Sensors: Theory, Algorithms, and Applications. Springer, 2012. 145–163.

[121] Wang, Jianxin, Peng, Wei, and Wu, Fang-Xiang. "Computational approaches to predicting essential proteins: A survey." PROTEOMICS-Clinical Applications 7 (2013).1-2: 181–192.

[122] Williams, Billy M and Hoel, Lester A. "Modeling and forecasting vehicular traffic flow as a seasonal stochastic time series process." Tech. rep., 1999.

[123] Zachary, W.W. "An information flow model for conflict and fission in small groups." Journal of Anthropological Research 33 (1977): 452–473.

[124] Zheng, Chan, Zhang, Yiqing, and Yin, Ling. "Constructing (k, r)-connected dominating sets for robust backbone in wireless sensor networks." Communications and Information Technologies (ISCIT), 2011 11th International Symposium on. IEEE, 2011, 174–177.

[125] Zhou, Zongheng, Das, Samir, and Gupta, Himanshu. "Connected k-coverage problem in sensor networks." Computer Communications and Networks, 2004. ICCCN 2004. Proceedings. 13th International Conference on. IEEE, 2004, 373–378.

BIOGRAPHICAL SKETCH

Chrysafis Vogiatzis was born in Athens, Greece in 1984. He received his diploma of engineering degree (equivalent to M.Sc.) from the Aristotle University of Thessaloniki in Electrical and Computer Engineering in 2009. His thesis on computational methods and computer science was entitled "Iterative distributed decomposition algorithm for large-scale transportation problems". In January 2010, Chrysafis Vogiatzis joined the graduate program of Industrial and Systems Engineering in the University of Florida, where he received a M.Sc. in 2012 and a Ph.D. in 2014.

During his studies in the University of Florida, he was recognized for excellence in teaching from the Department of Industrial and Systems Engineering and the University of Florida, when he earned the prestigious University of Florida Graduate Student Teaching Award. His research interests include, but are not limited to, mathematical programing and optimization, graph theory, applied mathematics, and operations research. His work has been published in journals of Springer, Elsevier, ASCE, MDPI, and the International Research Committee on Disasters and he has served as a co-editor for two volumes for Springer. Further, he is at the moment authoring a book on Multidimensional Assignment Problems and their applications. He is a student member of the Institute of Operations Research and Management Science (INFORMS) and the Society of Industrial and Applied Mathematics (SIAM).