UNIVERSIDADE DO ALGARVE

Route Planning in Wireless Sensor Networks for Data Gathering Purposes

Andriy Mazayev

Master Thesis in Computer Science Engineering

Work done under the supervision of: Prof[a] Noélia Correia (FCT/DEEI) e Prof[a] Gabriela Schütz (ISE/DEE)

2015

Statement of Originality

Route Planning in Wireless Sensor Networks for Data Gathering Purposes

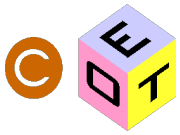
Statement of authorship: The work presented in this thesis is, to the best of my knowledge and belief, original, except as acknowledged in the text. The material has not been submitted, either in whole or in part, for a degree at this or any other university.

Candidate:

_____

(Andriy Mazayev)

Copyright ©Andriy Mazayev. A Universidade do Algarve tem o direito, perpétuo e sem limites geográficos, de arquivar e publicitar este trabalho através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, de o divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.



NETWORKING

Work done at Research Center of Electronics Optoelectronics and Telecommunications (CEOT)

Abstract

Recent studies have shown that mobile elements (ME) in wireless sensor networks (WSN) can drastically reduce energy consumption in such type of networks and thus prolong the sensor network lifetime. Most recent mobile elements used for data gathering are called unmanned aerial vehicle (UAV), commonly known as drones. Although these elements are far faster than terrestrial mobile elements, and used in a wide range of situations, there is a small number of studies on the effectiveness of their use.

In this thesis, after making an analysis of some of the solving methods that might be useful for data gathering, an algorithm is proposed that can provide an efficient use of UAV elements in WSNs. A performance analysis and comparison with existing approaches is done. Results show that the proposed algorithm is able to solve the data gathering problem efficiently presenting, for most data sets and time delivery deadline ranges, a high certainty degree regarding the quality of the solution.

Keywords: mobile elements, wireless sensor networks, routing, optimization

Resumo

As redes de sensores têm vindo a ganhar importância em vários setores da nossa sociedade, sendo a aplicação mais visível a monitorização ambiental. Estas redes têm inúmeras aplicações que vão desde a detecção de fogos, medições do nível de emissões de radiação, detecções de danos estruturais e muito mais, sendo capazes de fazer medições de forma sistemática, e possibilitando desta forma, tomar decisões antecipadas e, assim, evitar acidentes que podem custar vidas humanas.

O envio de dados recolhidos pelos sensores é tradicionalmente feito através de múltiplos saltos (*multi hop*). Este método baseia-se na passagem dos dados de dispositivo para dispositivo, através da infraestrutura da rede, permitindo que a informação viage desde a fonte até ao destino graças à passagem dos dados entre elementos vizinhos. Para encontrar a forma mais eficiente e rápida de fazer chegar a informação ao destino foram propostos, e estão disponíveis na literatura, vários algoritmos que tentam garantir o uso eficaz dos recursos da rede durante o envio dos dados. No entanto, nem mesmo os algoritmos mais eficientes são capazes de resolver um problema inerente a esta arquitectura: a entrega dos dados está dependente de vários elementos intermediários. Para além disso, o próprio meio ambiente em que as redes de sensores são utilizadas pode ser um obstáculo durante a comunicação. Possíveis acidentes tais como fogos ou desabamentos podem danificar os canais de comunicação e, desta forma, tornar uma rede de sensores inutilizável. As desvantagens presentes na abordagem clássica de recolha de dados levou os investigadores a procurarem outras alternativas.

Estudos recentes têm mostrado que a utilização de elementos móveis em redes de sensores sem fio pode reduzir drasticamente o consumo de energia neste tipo de redes e, assim, prolongar o seu tempo de vida. As abordagens mais recentes no processo de recolha dos dados numa rede de sensores têm recaído sobre o veículo aéreo não tripulado, frequentemente conhecido como drone. Embora estes elementos sejam mais velozes do que os elementos móveis terrestres, e devido à sua natureza que permite a sua utilização numa vasta gama de situações, há um pequeno número de estudos sobre a eficácia da sua utilização.

Ao contrário da abordagem multi salto, onde os algoritmos procuram caminhos eficientes dentro da própria infraestrutura de rede, a abordagem que envolve elementos móveis necessita de algoritmos que criem caminhos para os próprios elementos móveis. O processo de desenho e criação das rotas é conhecido por problema de recolha de dados

(*Data Gathering Problem*). O problema de recolha de dados a ser resolvido nesta tese pode ser descrito como o processo de criação de caminhos para um conjunto de veículos aéreos não tripulados que têm de recolher dados alojados nos sensores. Terão que ser considerados os seguintes aspectos:

- Janelas temporais de cada sensor. Intervalo de tempo em que o drone tem que visitar o sensor em causa;
- Volume dos dados a serem enviados pelos sensores;
- Tempo de vida útil dos dados. Intervalo de tempo máximo permitido para que a informação seja entregue no destino;
- Conjunto de veículos aéreos. Cada veículo aéreo tem a sua capacidade de memória, isto é, limite máximo da quantidade de dados que um drone é capaz de recolher.

O processo de criação dos caminhos tem que respeitar todas as restrições temporais dos sensores e, ao mesmo tempo, terá que ser minimizada a distância necessária para recolher os dados de todos os sensores.

A utilização dos veículos aéreos não tripulados é uma abordagem recente e, talvez por isso, ainda não ganhou a merecida atenção. Como tal, de momento, não existem standards de medição de desempenho de diferentes algoritmos capazes de resolver este tipo de problemas. Nesta tese, tentamos preencher esta lacuna propondo um padrão de medição de desempenho que pode ser utilizado por todos.

São apresentadas duas formalizações matemáticas do problema, uma genérica que pode ser utilizada em vasta gama de situações com semelhantes restrições e uma formalização estendida que representa o problema a ser resolvido nesta tese. A formalização genérica permite, identificar este problema com o do encaminhamento de veículos com janelas temporais (*Vehicle Routing Problem with Time Windows*).

É feita ainda uma análise aprofundada de algumas classes de algoritmos já existentes que podem ser úteis na resolução deste tipo de problemas. Heurísticas construtivas, de melhoramento e meta-heurísticas são introduzidas e alguns dos seus principais métodos são discutidos. Com base nestes métodos foi desenvolvida uma heurística híbrida capaz de criar caminhos eficientes não só para o problema recolha de dados numa rede de sensores mas também capaz de resolver o problema de encaminhamento de veículos. A heurística apresentada é baseada em três etapas de optimização. A primeira, pré-optimização, que com base nos dados de entrada, cria uma boa solução parcial que será utilizada nos passos seguintes. A segunda fase, designada por optimização, que recebe a solução parcial criada no passo anterior, é responsável pela construção de uma solução inicial que satisfaça todas as restrições de entrada. A última fase, pós-optimização, que recebe como dado de entrada a solução obtida durante a optimização, através de uma combinação de métodos

tanto determinísticos como não determinísticos explora de forma eficiente o espaço de procura em busca da melhor solução.

Para medir o desempenho da heurística híbrida desenvolvida nesta tese foi feito um extensivo conjunto de testes considerando de redes de sensores de várias dimensões, redes com 25, 50, 75 e 100 sensores, e diferentes níveis de restrições temporais. Os resultados obtidos mostram que o algoritmo proposto é capaz de resolver o problema de recolha de dados de forma bastante eficiente apresentando, para a maioria dos conjuntos de dados e para diferentes intervalos de prazos de entrega, um alto grau de certeza relativamente à qualidade da solução.

Termos chave: elementos móveis, redes de sensores sem fio, encaminhamento, optimização

Acknowledgements

I would like to thank my parents for their encouragement, education, for their belief in me and for their effort to make my graduation possible. Also, I am grateful to my sister for providing me support during all this time. And finally, I would like to thank my little niece for making my days brighter with her smile.

Special thanks to all my friends that have been by my side since the beginning of this journey. Thanks for your support, all the funny moments that we lived and just for being there for me when I needed it. Without you nothing of this would have been possible.

In academic environment I would like to thank my supervisors, Noélia Correia and Gabriela Schütz. Without their support and guidance I would not have been able to finish this project. Their experience and advices were invaluable.

Thank you all.

Contents

List of Figures

List of Tables

List of Algorithms

List of Publications

1. Pedro J.S. Cardoso, Gabriela Schütz, Andriy Mazayev, Emanuel Ey, ”Solutions in under 10 seconds for vehicle routing problems with time windows using commodity computers”, 8th International Conference on Evolutionary Multi-Criterion Optimization, EMO 2015; Guimarães, Portugal, 29 March – 1 April 2015.

2. Pedro J.S. Cardoso, Gabriela Schütz, Jorge Semião, Jânio Monteiro, João Rodrigues, Andriy Mazayev, Emanuel Ey, Micael Viegas, Carlos Neves, Sérgio Anastácio, ”Integration of a food distribution routing optimization software with an enterprise resource planner”, International Conference on Geographical Information Systems Theory, Applications and Management, Barcelona, Spain, 28-30 April, 2015.

3. Pedro J. S. Cardoso, Gabriela Schütz, Andriy Mazayev, Emanuel Ey, Tiago Corrêa, ”A Solution for a Real-time Stochastic Capacitated Vehicle Routing Problem with Time Windows”, 15th International Conference on Computational Science, Reykjavík, Iceland, 1–13 June (accepted).

4. Andriy Mazayev, Noélia Correia, Gabriela Schütz, ”Heuristic Approach for Data Gathering in Wireless Sensor Networks”, IEEE International Conference on Communications, London, United Kingdom, 8–12 June (accepted).

CHAPTER 1

Introduction

1.1 Background

Sensor networks are ad-hoc networks composed of sensors that are connected wirelessly. The main goal of sensors it to capture the state of the environment (temperature, humidity, luminosity, etc.) and to relay this information to an entity (sink node or base station) more suited for data processing, visualization, analysis and decision making upon the information received. A sensor node, also called node, is a small sensing capable device with limited processing capability, reduced memory capacity and bandwidth, and a limited source of energy (Rodrigues and Neves, 2010). Despite of all these hardware limitations, sensors are very useful nowadays. They are cheap, small in size, easy to install and highly flexible. This allows them to be used in a high range of situations, from simple luminosity measurements to radiation measurements in atomic plants (Chin et al., 2010).

Since sensor nodes are limited in processing capabilities, local data analysis may not be a viable option and, therefore, it becomes crucial to forward the data to a device with higher capabilities for further processing. Usually this is done by the network infrastructure itself without any third part involved in this operation. This type of data forwarding is called multi-hop and it involves one or more nodes to forward the data from a sensor to a sink node. In such cases, a power consumption problem arises. In order to send the data from one point to another, other nodes must first receive the information and then forward it, even without having anything do to with the data itself. It is known that, in sensors networks, the transmission process is the one spending more energy. Hence, the lifetime of a network is reduced if data is simply forwarded with no energy concern in mind. Replacing the energy source of a sensor may not be viable because of the surrounding conditions where the sensors are installed (atomic plant, underground, etc.). In most cases once the sensor is discharged there is nothing left to be done. All these constraints mean that data transmission must be thoughtful and must be used only when necessary in order to extend network operation.

Another and more recent approach to gather the data in sensor networks involves mobile elements. By moving directly to the source, receiving the information directly from

it and then delivering it to the destination, mobile elements allow multi-hop communication, which is energetically inefficient, to be avoided and thus the lifespan of a network to be extended. A mobile element can be of terrestrial type, travelling over the land, or an aerial element, called unmanned aerial vehicle (UAV).

This thesis focus on data gathering in WSNs using elements of UAV type. The goal is to develop an algorithm that can design a set of efficient paths to gather the data generated by sensor nodes, which are scattered over a given area. More specifically, the contributions of this thesis are:

- A mathematical formalization of the data gathering problem is presented;
- A deep analysis of the existing solving methods, useful in these kind of problems, is done;
- A hybrid heuristic algorithm is proposed;
- A standardized benchmark is introduced to measure the performance of the proposed algorithm;
- A head-to-head comparison between the results obtained by the proposed algorithm and the optimal solutions is done

1.2 Data Gathering Approaches

As stated earlier there are two approaches to gather the information in a sensor network. Each of these has it own pros and cons that will be discussed in the following subsections.

1.2.1 Classical Networks

In classical networks the data is transferred through the network infrastructure itself from the sensor node to the base station. Due to energy concerns the throughput in a sensor network is very limited, normally reaching only a few Kbps. Transmission is the most energy demanding task that sensor nodes perform. This means that any communication, and specially multi-hop communication, must be avoided and used only when it is strictly necessary to maintain network operation for a long time. The main advantage of this approach is the small delay on data delivery, which can reach a few seconds or even less.

1.2.2 Networks with Terrestrial Elements

In this approach mobile robots are used to collect the data from sensors for a later delivery to a base station for further processing. In this case bandwidth is higher, capable to achieve

few Mbps. The most prominent issue with this approach is a time delay between receiving the data from all sources and delivering it to destination. Since the travelling speed of terrestrial elements is considerably low, in most cases inferior to one meter per second (Tekdas et al., 2009) (Richard Pon, 2005) the delay between sensing the data at the sensor and processing it at the base station is significant. It may take several hours to gather all the data and deliver it to a base station.

Similarly to the previous approach a power consumption problem also arises. Power consumption at mobile elements is very high since mobile elements may have considerable size and must visit all the nodes to gather the information. However, in this case, replenishing the power source of the mobile element is not a problem since it is easily accessible. This approach can fully eliminate multi-hop communication at the expense of delay.

1.2.3 Network with UAV Elements

A UAV approach is in all similar to the approach stated above. The main advantage in this case is the drastic reduction of the delay. Since we are talking about an element with flying capabilities its travelling speed is much higher than terrestrial elements. Therefore, it can collect data scattered across all the sensors much more efficiently and quickly. Some research has been done on the use of UAVs in WNS (Sujit et al., 2013), (Pignaton de Freitas et al., 2010), (Costa et al., 2012). However, the UAV approach has still not reached the deserved attention.

Performance metrics	Multi-Hop	Terrestrial Element	UAV
Delay	Low (second)	High (hours)	Medium (minutes)
Energy Consumption	High (non rechargeable)	High (rechargeable)	High (rechargeable)
Bandwidth	Medium (few Kbps)	Medium-High (few Mbps)	Medium-High (few Mbps)

Table 1.1: Comparison between different data gathering approaches.

1.3 Network Design Problems in Sensor Networks

1.3.1 Topology Design

Designing an efficient sensor network may be a problem by itself. How to know where to put the sensors in order to achieve a good performing network? What to do if sensor nodes are scattered or randomly placed? The physical distribution of nodes can be a difficult

task by itself. In case of random sensor distribution there is no guaranty whatsoever that multi-hop based protocols will work. Some of the sensor nodes may be out of range of the network thus compromising its quality. A network should always be carefully designed and optimized in order to work properly, which usually involves environment analysis and studies. Different topologies have been proposed to maintain the reliability of the sensor network (Mamun, 2012). In a star topology the sensor nodes maintain a direct communication line with the base station. This basic approach is suitable for simple cases. However, due to direct communication limitation this topology in not capable to cover extensive areas and requires predetermined positioning of nodes. A tree topology is capable to solve this coverage issue. As in a star topology, each node has a single communication path toward the base station but in this case other nodes can act like forwarding elements and thus the covered area is extended. However, the topology is not fault tolerant because if a single routing node fails the sensor nodes will lose their communication path toward the base station. The mesh topologies, by adding redundant communication paths, remediate the reliability issues of tree topologies at the expense of additional hardware.

The use of mobile elements that can go directly to a source and gather the information could greatly relief the process of topology design because of routing flexibility.

1.3.2 Network Layer

Even if the topology is well designed there are still issues with sensor networks. How to correctly disseminate the data? How to efficiently forward data provenient from another node? The layer responsible for creation of paths between a source and a destination is called network layer. This layer is strongly influences the efficient use of network infrastructure. The process of data delivery through the network is called routing. In case of predetermined node distribution it is possible to create delivery paths *a priori*. However, in a random distribution that can result in non uniform topologies the sensors must be able to determine their positions, discover their neighbours and identify paths to a base station, which can be a difficult task.

Another critical issue in wireless sensor networks are choke points (Figure 1.1). As shown in Figure 1.1 a sensor node where choking occurs not only has to send its own data to a base station but also has to forward the data from many other nodes. As a result the power consumption of this node will be extremely high and it will deplete its power source much quicker than the other sensor nodes. When this occurs the network becomes unusable since the information does not reach its destination. Moreover, if a sensor network is composed of sensor nodes of different types and different sampling rates then network traffic becomes unbalanced resulting in performance inefficiency. Paths must be designed in a balanced way in order to extend the life span of a network. To solve

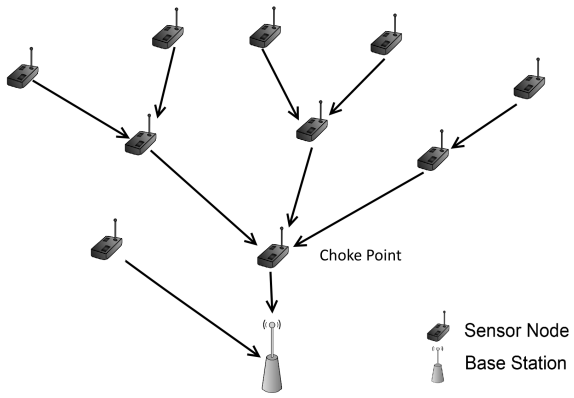


Figure 1.1: Choke point.

the issue presented in Figure 1.1 traffic could be split between two sensors nodes that are in the vicinity of the base station.

Another approach to deliver the data can be achieved with a mobile element. In this case, instead of designing paths for a multi-hop communication, the paths are designed for the mobile element itself. This way, the mobile element travels throughout all the network in an efficient way and gathers all the information directly from the source before delivering it to a base station. An optimization of paths is a crucial point of a good performing sensor network.

After exposing all these issues it is now possible to generally describe the problem to be solved in this thesis.

1.4 Problem Definition

Given a set of nodes distributed on a surface with respective coordinates and their time restrictions, given a sink node with its own coordinates and also a set of UAVs that can freely travel over the surface, the goal is to create paths in such a way that:

- All node are visited only once
- All routes begin and end at the sink node
- All time restrictions are respected
- The number of mobile elements is kept minimum during the data gathering process
- Total travelling distance is minimized

In this thesis it is assumed that the distance between any pair of nodes can be given by the euclidean distance, which may not be possible in reality but since we are trying to make a theoretical demonstration euclidean distance will suffice. Also, the distribution of

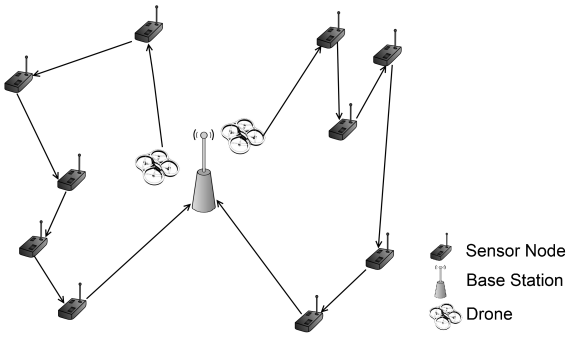


Figure 1.2: Problem illustration.

nodes may be random (R), clustered (C) or both random and clustered (RC). Finally, each sensor nodes has its own unique time restrictions.

The problem described in this section is NP-hard. A more detailed description of this implication will be done in Chapter 2.

1.4.1 Practical Applications

Due to the existence of some delay when using mobile elements the approach presented in this thesis should not be used in critical situations, such as fire, radiation, structural damage monitorization and detection when it is required response at the time. In these cases every second counts and, therefore, the classical multi-hop approach is preferable. However, even in these cases, due to structural damage and possible disruptions in communication channels drones may become handy as they do not require additional infrastructure to gather the data. In other cases where the delay is not an important aspect mobile elements can and should be used.

At the research center where this thesis was developed, this problem, is expected to be applied, in the future, to detect the cork tree decease at an early stage. This project is just starting at the Center of Electronics Optoelectronics and Telecommunications (CEOT) and is currently in an early development stage.

The UAV approach can also be used to forecast eventual wildfire occurrences and to warn about the necessity to perform prevention measures. In order to accomplish this it is necessary to place temperature, humidity and wind sensors over the area being monitorized and periodically gather the data from them. In Portugal, especially during the summer, the wildfire is a common event and monitorization could reduce drastically the number of fire incidents.

CHAPTER 2

Mathematical Optimization

2.1 Introduction

Mathematical optimization is a field of mathematics whose goal, depending of the problem to be solved, is to minimize or maximize the value of the objective function through the combination of feasible variables. Generally, a mathematical optimization can be described in the following way:

$$min_{x \in S} f(x) \tag{2.1}$$

where:

- $f(x)$ is the objective function;
- S is the set of feasible values that x can take;
- x, depending of the problem, can be integer or real.

The expression 2.1 means that variable x can only take values of set A and for each of these values the objective function gives a quantitative description about the solution. In this case in particular, the goal is to find x in such way that the $f(x)$ is minimum.

The duality theorem allows to rewrite the function 2.1 as maximization problem. However, since we are tackling a cost minimization problem, the remaining part of this report will only focus on minimization functions.

2.2 Combinatorial Optimization

Combinatorial optimization is one of the branches of the mathematical optimization. Likewise the mathematical optimization, the goal is to minimize the cost of the objective function. However, in combinatorial optimization the variables are discrete and belong to a finite set.

For a better understanding of the combinatorial optimization concept let us take a look at a simple example.

Minimize the perimeter of a square in such a way that the product of length and width is equal to 10.

$$min 2x + 2y \tag{2.2}$$

subject to:

$$xy = 10 \tag{2.3}$$

$$x, y \in \mathbb{N} \tag{2.4}$$

Table 2.2 presents the set of possible solutions for the problem stated above.

Solution	x	y	Perimeter
1	1	10	22
2	2	5	14
3	10	1	22
4	5	2	14

Table 2.1: Solutions for perimeter minimization.

As shown in Table 2.2, all solutions are feasible since all of them satisfy the constraints. However, only solution 2 and 4 minimize the perimeter and thus only these two solutions suit this problem.

2.3 NP-Hardness

Routing and scheduling problems complexity have been largely explored and studied and it has been consistently proven that routing problems are NP-hard. NP problems are not solvable in polynomial time $O(n^k)$ where n is the input size of a problem and k is some constant. In fact, NP class consists of a set of problems whose solution can be efficiently verified but finding the actual solution is a quite difficult task.

The fact that the problem studied in this thesis is NP-hard implies that alternative approaches are required to solve it, namely heuristic methods. A more detailed explanation about routing and scheduling problems complexity can be found in (Lenstra and Kan, 1981).

2.4 Mathematical Formulation

A generic data gathering problem consists in collecting the data of a set of nodes using a set of UAVs. The problem can be represented by a graph $G = \{N, A\}$ where $N = \{0, 1, \ldots, n, n+1\}$ is a set of vertices, $A \subset \{(i,j) : i,j \in N\}$ is a set of edges and $C = N \setminus \{0, n+1\}$ is the set of sensor nodes (Ombuki et al., 2006).

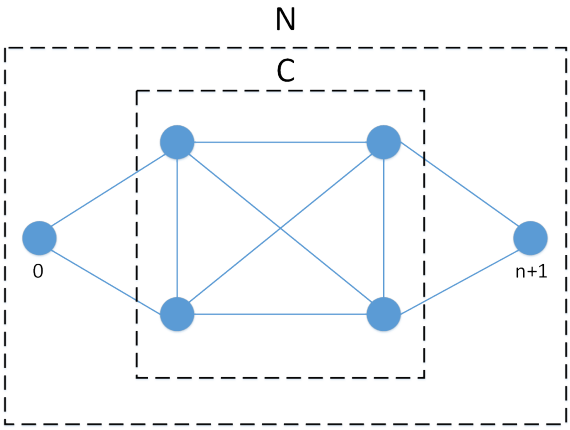


Figure 2.1: Graph representation.

Nodes 0 e $n+1$ represent the base station, starting and ending points for all UAV's paths. The remaining nodes $1, 2, ..., n$ represent the sensor nodes that must be visited. If in a real case the starting and ending points are the same then node $n+1$ can be seen as a virtual node for formulation purposes. The set of edges of a graph, represented by A, contains all possible link between all the nodes, including the base station, of the problem. In this case in particular it is assumed that the graph is complete, i.e., it is possible to travel from a node $n_i \in N$ to any other node from $n_j \in N \setminus n_i$

Each edge has its own cost c_{ij} and a time $t_{ij}, (i,j) \in A$. The cost c_{ij} represents the cost associated to a travel between the vertex i and vertex j. Similarly, t_{ij} represents the time required to travel from vertex i to vertex j. In this case, and for theoretical purposes, it is assumed that the $c_{ij} = t_{ij} = d_{ij}$ where d_{ij} is the euclidean distance between vertices i and j. But the developed algorithm is generic and can perform optimization even in cases where the equality is not met.

The set of UAVs is denoted by V and each UAV $v \in V$ has it own buffer size q_v. Each sensor node n, represented by a vertex at the graph, has data d_n to be gathered, $n \in C$. Each sensor node n can have a time window $[st_n, et_n]$, where st_n and et_n represent the start time and end time of a sensor node, respectively. An UAV can arrive to a node n before its start time but it will have to wait until st_i to receive the data. In the other hand, an UAV is not allowed to arrive to a node n after end time et_n.

All UAVs must start at the base station within the range $[st_0, et_0]$ and arrive to a base station within $[st_{n+1}, et_{n+1}]$. These intervals represent time windows of a base station.

In this particular case, the start time restriction is irrelevant because base stations are, normally, always active and we can assume that $st_0 = 0$, i.e., all routes start at instant 0. However, to design a generic algorithm that could operate even in cases when a base station has other time windows, this restriction should be taken into account.

The model described results in a problem with three decision variables, x, v and a. For each edge (i, j), where $i \neq j, i \neq n+1, j \neq 0$, and for each UAV v, the decision variable x_{ijv} is equal to 1 if UAV v travels from node i to node j, 0 otherwise. The decision variable a_{iv} represents the arrival time of UAV $v \in V$ to node $i, i \in C$. The decision variable ϑ_v is equal to 1 if current UAV is required to gather the data, 0 otherwise.

The goal of the gathering problem is to gather the data from nodes C by using UAVs from V in a way that the distance during the gathering process is minimized. This constraint can be expressed using a cost function because smaller distance implies smaller costs. However, these requirements must comply with the following constraints:

- Buffer size of UAV
- Time windows
- Each node must be visited only once
- All routes begin at node 0 and end at the node $n+1$

The problem is formulated as follows:

$$min \sum_{v \in V} \sum_{i \in N \setminus \{n+1\}} \sum_{j \in N \setminus \{0\}} c_{ij} x_{ijv} \tag{2.5}$$

Subject to:

$$\sum_{v \in V} \sum_{j \in N \setminus \{0\}} x_{ijv} = 1, \qquad \forall i \in C \tag{2.6}$$

$$\sum_{v \in V} \sum_{i \in N \setminus \{n+1\}} x_{ijv} = 1, \qquad \forall j \in C \tag{2.7}$$

$$\sum_{i,j \in sXs} x_{ijv} \leq |S| - 1, \qquad \forall v \in V, S \subseteq \{1, ..., n\}, 2 \leq |S| \leq n-1 \tag{2.8}$$

$$\sum_{i \in C} d_i \sum_{j \in N \setminus \{0\}} x_{ijv} < q_v, \qquad \forall v \in V \tag{2.9}$$

$$\sum_{j \in N \setminus \{0\}} x_{0jv} = \vartheta_v, \qquad \forall v \in V \tag{2.10}$$

$$\sum_{i \in N \setminus \{n+1\}} x_{ihv} - \sum_{j \in N \setminus \{0\}} x_{hjv} = 0, \qquad \forall v \in V, \forall h \in C \tag{2.11}$$

$$\sum_{i \in N \setminus \{n+1\}} x_{i,n+1,v} = \vartheta_v, \qquad \forall v \in V \tag{2.12}$$

$$a_{0v} = 0, \qquad \forall v \in V \tag{2.13}$$

$$a_{iv} + s_i + t_{ij} - K(1 - x_{ijv}) \leq a_{jv}, \quad \forall v \in V, \forall i \in N \setminus \{n+1\}, \forall j \in N \setminus \{0\}, i \neq j \tag{2.14}$$

$$\vartheta_v st_i \leq a_{iv} \leq \vartheta_v et_i, \qquad \forall v \in V, \forall i \in N \tag{2.15}$$

$$x_{ijv} \in \{0,1\}, \vartheta_v \in \{0,1\}, a_{iv} \geq 0, \quad \forall v \in V, \forall i \in N, \forall j \in N \tag{2.16}$$

Where:

Sets	$V = \{1, 2, ..., v\}$	set of UAVs
	$N = \{0, 1, ..., n, n+1\}$	set of nodes including the base station
	$C = \{1, 2, ..., n\}$	set of nodes
	$A = \{(i_1, j_1) : i, j \in N, i \neq j\}$	set of edges
Constants	d_i	amount of data to be gathered at node $i \in N$
	st_i	start of the time window of node $i \in N$
	et_i	end of the time window of node $i \in N$
	q_v	buffer size of UAV $v \in V$
	c_{ij}	cost of travelling from node i to j, $i, j \in N$
	t_{ij}	travel time from node i to j, $i, j \in N$
	s_i	service time at node $i \in C$
	K	is a large value
Decision variables	x_{ijv}	one if UAV $v \in V$ travels from node i to node j, $i, j \in N$
	a_{iv}	arrival time of UAV $v \in V$ to node $i \in N$
	ϑ_v	one if vehicle $v \in V$ is required to gather the data

The objective function (2.5) indicates that the costs must be minimized. Constraints (2.6) and (2.7) indicate that each node must be visited by a single UAV. Constraints (2.8)

avoid the formation of subcircuits. Constraints (2.9) indicate that buffer size of the UAV v can not be exceeded. Constraints (2.10), (2.11) and (2.12) are flow restrictions. The first one (2.10) indicates that all UAVs must start at the base station, the second one (2.11) indicate that UAV can not stay at the node, i.e., the UAV must arrive to a node, gather the data and depart. Constraints (2.12) indicate that paths must end at the base station. Constraints (2.13) indicate that all UAVs start gathering process at instant zero. Constraints (2.14) indicate that UAV v can not arrive at node j before $a_{iv} + s_i + t_{ij}$ if it travels from node i to node j. Constraints (2.15) state that time window restriction must be met. Finally, the (2.16) restrictions define variables.

Note that this formalization can be applied to many gathering problems. In the following sections this will be adapted to the data gathering problem in WSNs using UAVs, where data collected can also have a period of validity (Time-To-Live), which is not included in this formalization.

CHAPTER 3

Solving Methods

Most of the research on data gathering in WSNs uses adaptations or variations of the Vehicle Routing Problem (VRP) that was and continues to be a hot topic in the optimization field. The main reason is attached to economic grounds. More and more transport companies are embracing software utilities which can improve their business models in many aspects such as cost reduction and more efficient use of resources during the distribution and/or gathering of the goods. Another but not less important reason is the inexistence of a good performing algorithm which could solve all routing problems seamlessly. This simple fact draws the attention of researchers from all over the globe.

Mobile element routing have not draw much attention in academic field. As result, when compared with other optimization problems, such as the VRP, mobile element routing is just a drop in an ocean. Due to little research about mobile element routing in WSNs and due to its similarity with the VRP we will gather knowledge about solving methods in a known and largely studied area and adapt them to the mobile element routing.

3.1 State Of The Art

In this section we will take a look at some of the mobile element routing problems tackled by researchers.

In Data Gathering problem (Bhadauria and Isler, 2009) all sensor nodes must be visited in such way that the expended time to perform this task is minimum. This problem is a variation of k-Travel Salesman Problem.

Time-constrained Mobile Element Scheduling (Almi'ani et al., 2010) state that all sensor nodes must be visited within their own time window, i.e., period of time when sensor is awake and capable to communicate. This problem is a straightforward adaptation of VRP-TW in a sensor network.

In Deadline Scheduling (Somasundara et al., 2007) a set of sensor nodes with limited buffer size and different sampling rate must be visited before their buffer overflows. In Calibration Scheduling (Bychkovskiy et al., 2003) sensor nodes must be visited and calibrated before the error of sensed data exceeds a certain threshold. These problem can also

be viewed as VRP-TW where the deadline to visit a node represents a time window.

In Rendezvous Planning (Xing et al., 2008) there are intermediate buffer nodes (rendezvous points) where the data from the sensor nodes is stored. In this problem the data generated by sensors must be delivered to a base station within a certain deadline. The delivery is done by mobile elements that travel to rendezvous points to gather the data and deliver it to destination. This problem can be viewed as simplified VRP-PD (Pickup and Delivery) where the data must be picked up at rendezvous points and delivered to the destination.

As seen, all these are adaptations of VRP. Therefore we will take a close look at the original problem and its solving methods and then apply them to a WSN with mobile elements problem.

3.2 Vehicle Routing Problem

In the VRP the goal is to find the best set of routes to visit all the clients distributed across a certain geographical area. However, in order to understand this definition it is necessary to define clearly what is the ”best route”. Best route is one that results in a minimum usage of resources without violating any of restrictions imposed by the problem. Usually VRP has time and capacity restrictions.

- Time Restriction (Time Windows) - the vehicle must visit the clients in a time window defined by clients. Time windows can correspond to a normal working day. For example, let us assume that there is a client with the following time window - from 9 a.m. until 5 p.m.. This means that the distribution vehicle is obliged to visit and serve this client during this time window. In case of a mobile element routing in sensor networks this could correspond to a state when the sensor is in awake mode and is performing some activity. Similarly to a vehicle in VRP, the UAV must visit a sensor when it is active and has some data to send, or due to a deadline to visit a node with a limited buffer size.

- Restriction Capabilities (Capacitated Problem) - this restriction limits the number of visiting clients. A vehicle can not serve a client if it does not have the capacity to transport the required goods. In sensor networks with mobile elements this could correspond to data to be gathered by the UAV. An UAV can not visit and gather the data of a sensor node if its buffer does not have the capacity to receive the data.

3.3 Two Step-Algorithm Approaches

By nature, the human being has difficulty to process large quantities of information. Usually this task is done through the decomposition of the data into smaller and independent

parts followed by processing and analysis of those parts. This strategy is called divide-and-conquer. The division step implies the decomposition of a problem in small parts and conquer stands for a resolution of decomposed parts which, when combined, result in a solution to the original problem. This simple method has large applications in science, and computer science is no exception, not even the most powerful computers are able to solve, in a reasonable time, optimization problems with extremely large number of variables. In order to accelerate the search for the solution a two step algorithm can be used, Cluster First - Route Second.

3.3.1 Clustering First

Clustering is done in the initial phase, before solving the problem. At this stage all nodes (clients or sensor nodes) are equal and belong to a single set (Figure 3.1).

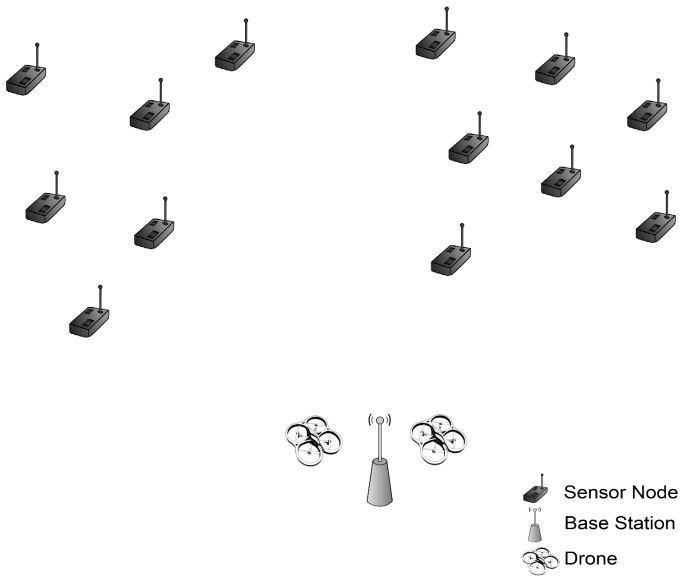


Figure 3.1: Example of a problem before clustering process.

The clustering process is done according to some criteria, normally by geographical localization but it also could be done following any other criteria that can allow the cluster formation based on the similarity features of the nodes (Figure 3.2).

Upon the completion of clustering stage, the original problem is decomposed in a set of independent sub-problems with a smaller size and easier to solve. Normally, each cluster can be seen as a travelling salesman problem (TSP) (Baker, 1983) to which time windows restrictions can be added.

3.3.2 Routing Second

Once the clustering is complete and the problem is divided, each of the clusters can be separately solved. For each cluster, a mobile element or a set of mobile elements will

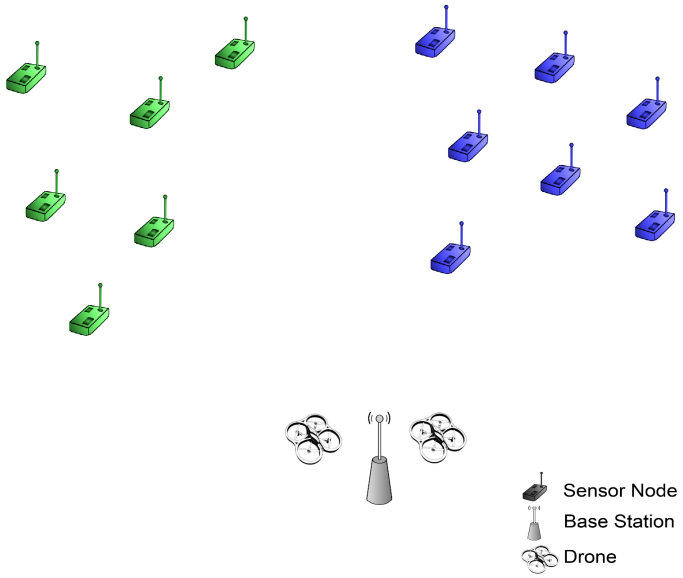


Figure 3.2: Example of a problem after clustering.

be assigned. They will be responsible for the data collection from their cluster (Figure 3.3). During this step a numerous amount of methods can be applied, exact approaches or heuristic and meta-heuristics.

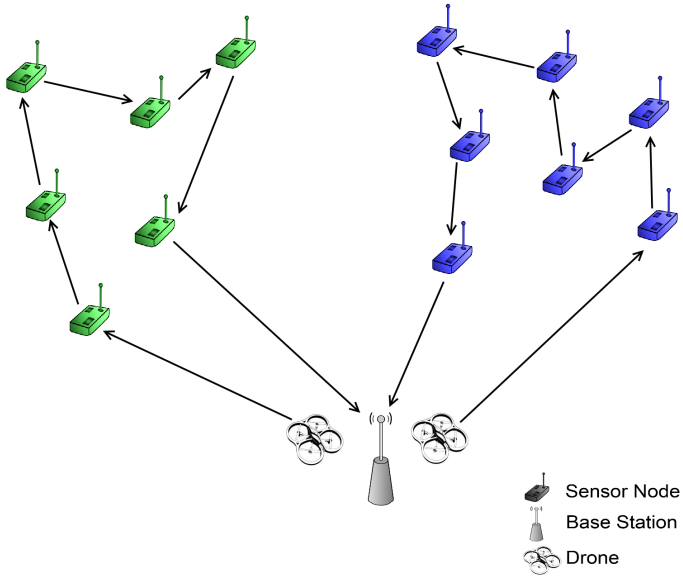


Figure 3.3: Example of a problem solution.

The two step algorithm facilitates the search of the solutions as it divides the problem in a set of independent sub-problems. The division of the original problem reduces drastically the search space, which consequently results in a more efficient search of the solution for the original problem.

3.4 Heuristics

Heuristics are a set of algorithmic methods whose goal is to find the optimum solution for an optimization problem. These methods have a certain degree of ”knowledge” about the problem, which allows them to perform a relatively small search in the search space. Usually heuristics are capable to generate acceptable solutions in short periods of time.

However, it is important to mention that there is no guarantee that the solution found is optimum or even if it is near it. The only way to evaluate the quality of the generated solution is to solve the problem and find the optimum solution but, as seen earlier in Chapter 2.3, this is a difficult task.

3.4.1 Constructive Heuristics

Constructive heuristics build the solution in an iterative way. These heuristics start with an empty solution and at each iteration the current solution is extended. This process is repeated until the complete solution is constructed. Usually constructive heuristics are greedy and do not possess backtracking mechanics, i.e., once an element is inserted in the solution it is impossible to remove it.

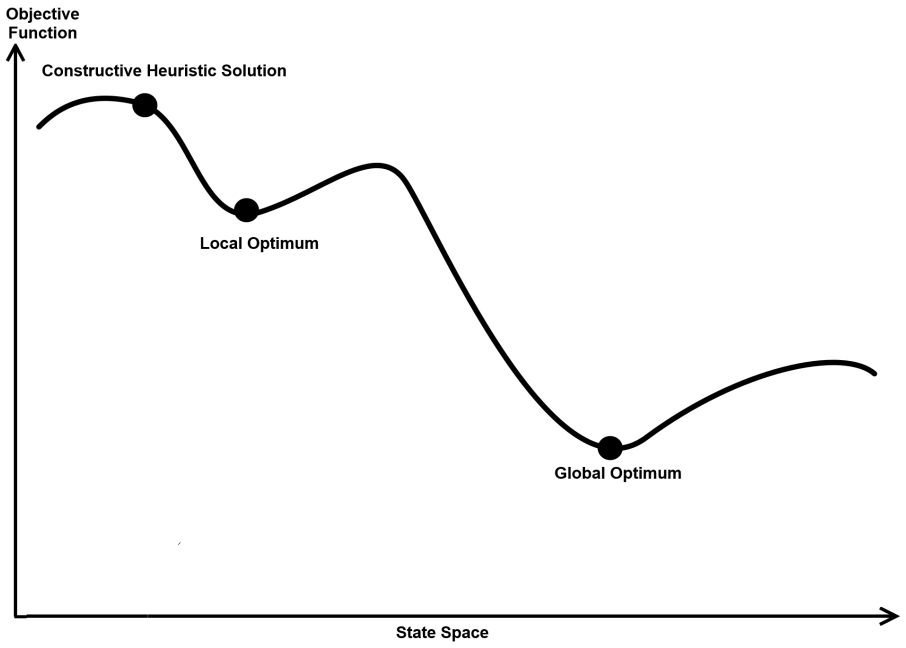


Figure 3.4: Solution produced by constructive heuristic.

Figure 3.4 is an example of what a constructive heuristic can produce. As shown, the solution generated can be far from optimum and could offer a lot of space for improvements. Improvements that can be done by other set of heuristics, which will be addressed in Chapter 3.5.

3.4.2 Nearest Neighbour Heuristic

Nearest neighbour heuristic is probably the most intuitive and, perhaps because of that, it is the easiest to understand. As the name suggests, the idea behind this algorithm is to choose always the closest node to the current position of the mobile element. However, a possible time restriction must be taken into account, meaning that choosing the closest node is not a straightforward process. In order to find the closest neighbour the following cost function can be used (Jaheruddin, 2010):

$$c_{ij} = \delta_1 d_{ij} + \delta_2 t_{ij} + \delta_3 v_{ij} \tag{3.1}$$

Where:

- c_{ij} travelling cost between node i and j
- d_{ij} distance between node i and j
- t_{ij} time difference between completing node i and staring node j
- v_{ij} remaining time until last possible start of j following i
- $\delta_1 + \delta_2 + \delta_3 = 1$ and $\delta_1 \geq 0, \delta_2 \geq 0, \delta_3 \geq 0$

The nearest neighbour heuristic can be described in a generic way by Algorithm 1.

Algorithm 1: Nearest neighbour heuristic.

input : $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ and set of UAVs $\mathcal{V}$.
output: Feasible solution

```
Start with first mobile element at the base station
while there are nodes to be visited do
    while mobile element is not full or there are nodes to visit do
        Find closest and unvisited node to current node
        Assign closest node to the current mobile element
    end
    Choose next mobile element
end
```

3.4.3 Clarke and Wright Savings Heuristics

The main concept of this method is to reduce costs by merging two or more routes in a single one (Florêncio, 2011). However route merging must comply to all the restrictions imposed by the problem being solved.

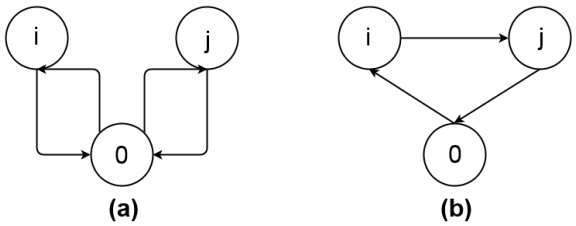


Figure 3.5: Example of route merging between node i and j.

The economy of resources by route merging can be quantified by a mathematical formula. For a better understanding of the formula let us apply it to Figure 3.5. In case (a) the travelling cost can be expressed in a following way:

$$d_a = c_{0i} + c_{i0} + c_{0j} + c_{j0} \tag{3.2}$$

Where c_{ab} is the cost of travelling from node a to a node b. Similarly for case (b):

$$d_b = c_{0i} + c_{ij} + c_{j0} \tag{3.3}$$

The amount of saved resources is given by the difference between the resources used in case (a) and in case (b)

$$S_{ij} = d_a - d_b = c_{i0} + c_{0j} - c_{ij} \tag{3.4}$$

However, before merging the two paths it is necessary to check if the resulting route will be able to satisfy all the restrictions imposed by the problem. There are cases where one route can be merged with two or more other routes. In this case, the tie is broken by the value of S_{ij}, route with the highest value of S_{ij} is chosen, as the goal is to minimize the costs. A generic description of Clarke and Wright savings heuristic is done in Algorithm 2.

The Clarke and Wright algorithm has two variations, the sequential one and parallel one. In the parallel version multiple routes are built simultaneously. Usually, the parallel version produces sightly better results.

3.4.4 Push Forward Insertion Heuristic (PFIH)

Push forward insertion heuristic (PFIH) is a greedy heuristic presented by Solomon (1987). This method has been implemented and tested by several authors (Thangiah et al., 1993), (Russell, 1995), (Thangiah, 1999), (Tan et al., 2001). The operational sequence of this algorithm is quite simple. First step is to classify and define the sequence by which nodes will be inserted in the solution. Usually, this is done by the following cost function

Algorithm 2: Clarke and Wright savings heuristic.

input : $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ and set of UAVs $\mathcal{V}$.
output: Feasible solution

```
Create a direct route for each node
while cost saving merge exists do
    Choose a route
    Compute all possible cost saving merges for current route
    Sort all merges by S_ij
    while current route is mergeable do
        if merge does not violate restrictions then
            Merge routes
        end
    end
    Next route
end
```

Thangiah et al. (1994):

$$c_i = -\alpha \times d_{i,0} + \beta \times et_i + \gamma((p_i/360) \times d_{i,0}) \tag{3.5}$$

Where:

- c_i PFIH cost of node i.
- $d_{i,0}$ distance between node i and base station.
- et_i upper limit of time window of node i.
- p_i polar angle between node i and the base station.
- $\alpha + \beta + \gamma = 1$

Different α, β and γ values can be used to give more or less importance to formula parcels, resulting in distinct ordering of the nodes. For instance, large values of α will make more preferable nodes farther from the depot. Larger values of β will make nodes with earlier closing window preferable. Larger values of γ will make preferable to insert the nodes in a circular spanning mechanism (like a "radar"). Normally, α is set to 0.7, β to 0.1 and γ to 0.2. However, these values were empirically established by Thangiah et al. (1994), Ghoseiri and Ghannadpour (2009) and other values can be used. Given the cost of all nodes, insertion is quite straightforward (Algorithm 3).

Usually this method is used to establish the upper bound of the problem. The solution produced by this heuristic is mostly used as input information in other methods for further improvements.

Algorithm 3: Push forward insertion heuristic.

input : $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ and set of UAVs $\mathcal{V}$.
output: Feasible solution

```
1  Calculate PFIH cost for each node
2  Sort all the nodes by their PFIH cost
3  while there are nodes to be visited do
4      if number of routes != 0 then
5          Insert in the route where the insertion causes minimal distance increase
6          if not possible then
7              Create new route
8              Insert current node into recently created route
9          end
10
11     else
12         Create new route
13         Insert current node into recently created route
14     end
15
16 end
```

3.5 Improvement Heuristics

Unlike constructive heuristics that start with an empty solution and build a new solution in an iterative way, improvement heuristics start with an already existing and feasible solution. At each iteration the improvement heuristic explores the neighbourhood of the current solution to find a better one.

The main problem with this process and with all improvement methods in general is that they do not possess mechanisms to escape local optimums. When a local optimum is found, the algorithm enters in a stagnation stage, i.e, all neighbours are worse than the current solution and no further improvement is possible. Figure 3.6 represents the search direction of the improvement heuristic, the sequence of the explored neighbours and the solution produced by it.

In this example the improvement heuristic received as input the solution produced in Figure 3.4.

3.5.1 Ruin and Recreate Principle

Since the constructive heuristics are usually greedy methods, i.e., at each iteration the decision is made by a local criteria, the final solution may be far from optimum. Hence solutions produced by constructive heuristics offer room for improvements most of the times. The ruin and recreate principle (Schrimpf et al., 2000) is a two step improvement method that can be used to improve current solution. During the first stage of this method

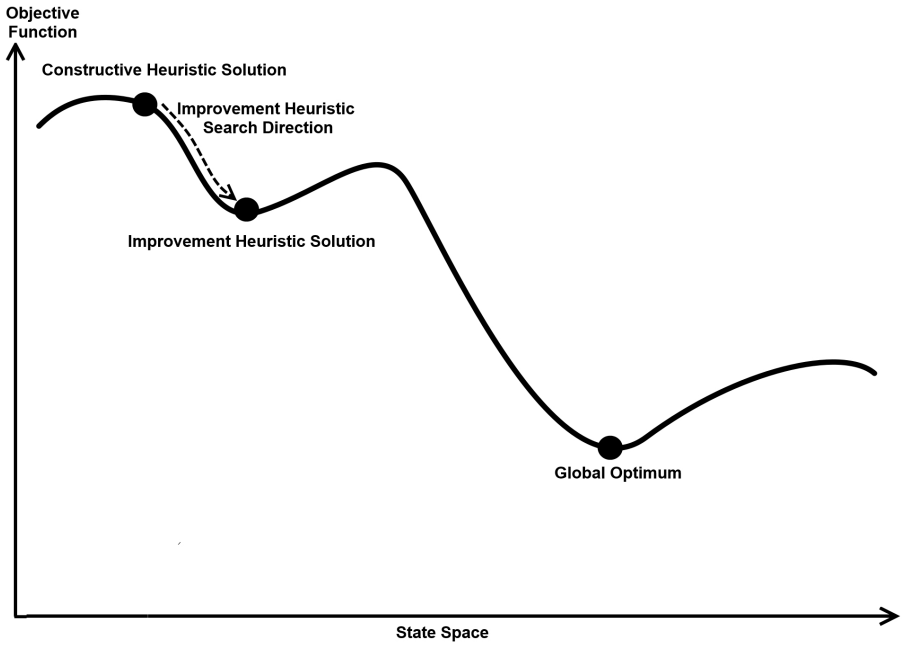


Figure 3.6: Solution produced by improvement heuristics.

a set of nodes is ejected from the solution. There are many criteria to eject a set of nodes:

- Random ejection - from each route a random number of nodes is ejected;
- PFIH cost based ejection (obtained after execution of PFIH (Chapter 3.4.4) - from each route nodes are ejected by their PFIH;
- Proximity ejection - a node is selected and a set if its neighbours is ejected.

During the second stage, recreate principle, the solution is restored in best possible way. In other words, during this stage ejected nodes are reinserted into solution. Usually the reinsertion is done by constructive heuristics but other methods can be used. Despite of the simplicity of this method its systematic repetition could greatly improve the solution given as input.

3.5.2 Local Search

Local search is an improvement heuristic and as such it needs an initial solution as input. This set of methods perform a set of operations on a given solution aiming to improve it (Carić et al., 2008). Local search operators can be divided in two large sets - intra routes and inter routes.

Intra routes operators (Figure 3.7) perform operations on a single route at the time. These operators rearrange the sequence of the nodes in a route in order to make it more efficient. Some of the intra routes operators are:

- Relocate
- Exchange
- 2-Opt
- Or-Opt

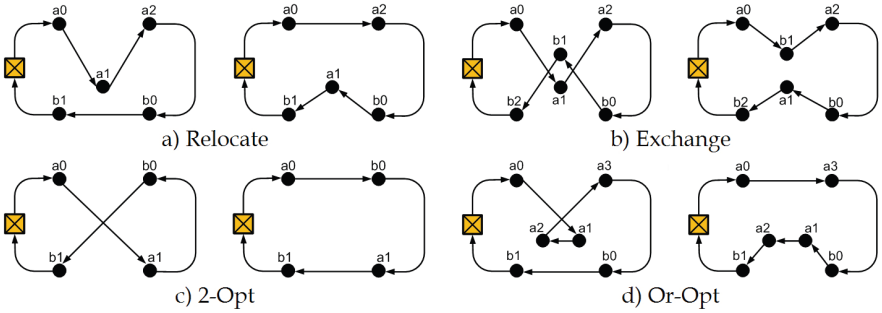


Figure 3.7: Intra route operators (Carić et al., 2008).

On the other hand, inter route operators (Figure 3.8) perform operations on two routes. These methods swap or rearrange the nodes between two routes in order to improve their quality and, consequently, to improve the solution. Some of the inter route operators are:

- Relocate
- Exchange
- Cross-Exchange
- Icross-Exchange
- 2-Opt*

Algorithm 4 describes generically the sequence of steps of the local search methods.

As stated earlier, improvement heuristics can stop when it is not possible to improve the current solution. Like the constructive heuristics there is no guarantee that the produced solution is the global optimum. Improvement heuristics do not possess mechanisms to escape local minimums and to explore new directions. Escaping mechanisms and techniques belong to a different set of heuristics, usually called meta-heuristics.

3.6 Meta-heuristics

Meta-heuristics are a set of procedures that combine heuristic and non-deterministic methods in order to perform an efficient search in the search space. Heuristic methods allow

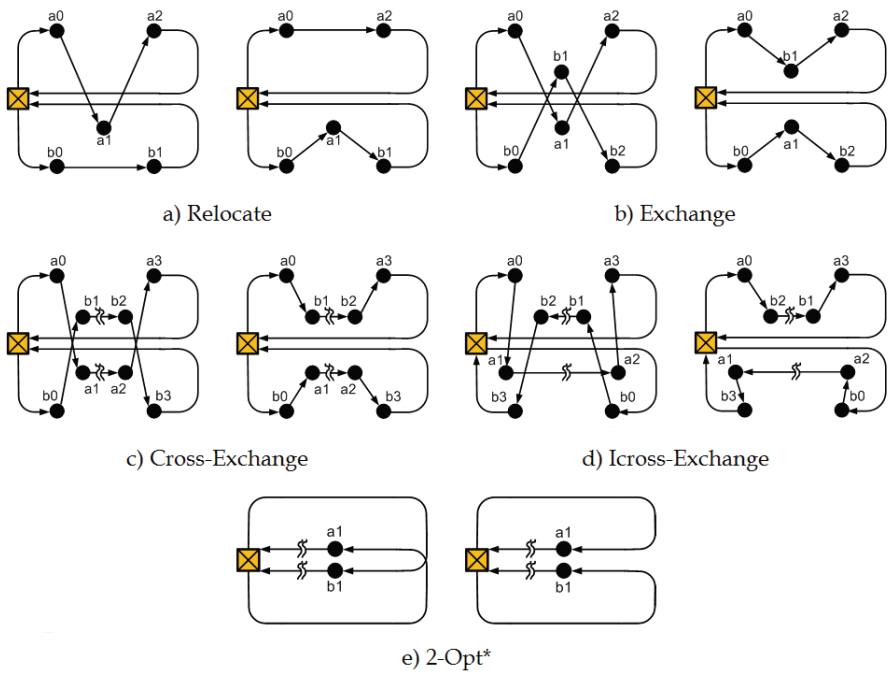


Figure 3.8: Inter route operators (Carić et al., 2008).

Algorithm 4: Local search.

input : Feasible Solution
output: Best solution found

```
while stopping criteria is not satisfied do
    Apply intra routes operators on current solution
    Apply inter routes operators on current solution
    if new solution is better than current solution then
        Replace current solution with new solution
    end

end
```

meta-heuristics to perform guided search to where a good solution might be. On the other hand, non-deterministic methods are used to escape local minimums and to explore new search directions. Usually meta-heuristics are used in combinatorial optimization problems, in which routing problems are included.

Figure 3.9 represents the application of meta-heuristics to the solution produced by improvement heuristics in Figure 3.6. In this example the solution found by the meta-heuristic is indeed the global optimum but actually, in real cases there is no guarantee of its optimality.

It is important to mention that the input given to a meta-heuristic can be produced by any other methods, it does not have to be necessarily produced by improvement heuristics. Figure 3.9 is just an example of a possible sequence of steps to find an optimum solution.

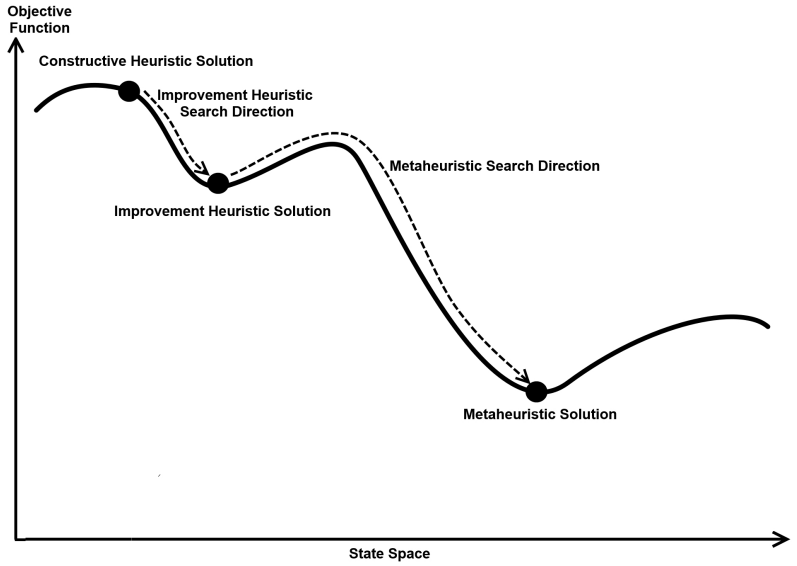


Figure 3.9: Solution produced by meta-heuristic.

3.6.1 Genetic Algorithms

Genetic algorithm is a meta-heuristic inspired on nature and on the way species evolve. According to the observations of Darwin, stronger species have greater probability to survive and to produce descendants at least as strong as their parents. In a similar way, weak species tend to produce weak descendants that tend to disappear over time. Genetic algorithms follow the idea described above and adapts it to solve numerous optimization problems.

Genetic methods systematically apply a set of operators (genetic operators) on a population (set of possible solutions) in order to find an optimal or a near optimal solution (Ombuki et al., 2006).

Algorithm 5: Genetic algorithm.

input : $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ and set of UAVs $\mathcal{V}$.
output: Best solution found

```
1 Generate initial population
2 while stopping criteria is not satisfied do
3 |   Apply selection operators on initial population
4 |   Apply crossover on a newly formed population
5 |   Apply mutation operators on population from crossover
6 |   Replace initial population with a new one
7 end
```

Genetic algorithms are easy to implement and when all of the parameters are properly tuned it can produce good results. However, tuning genetic algorithms is a difficult and a time consuming task, it requires large set of tests in order to discover the best combination

of parameters. Also it is important to notice that for different input the best combination of parameters may not be the same, meaning that it may be necessary to repeat the tuning process again. Another issue with genetic algorithms are the super individuals, i.e., solutions that are far better that the rest of the population. These solutions can quickly dominate the population and cause early convergence.

A more detailed description about different steps of genetic algorithms can be found in paper made by (Moura, 2008).

3.6.2 Tabu Search

Tabu search is an iterative meta-heuristic that performs an efficient space search by exploring neighbours of the solution given as input (Tam and Ma, 2008). The tabu search accepts neighbours that are slightly worse than current solution. The admissibility of an inferior solution may seem as a bad strategy, however this method allows to escape local minimums and to explore new search directions.

To avoid exploration in an already explored directions, all solutions generated by tabu search are placed, for a certain number of iterations, in a tabu list. Tabu list is a mechanism that makes the search more efficient and guided. This way, each time a new solution is generated it is necessary to check if it is already on the tabu list. If a newly produced solution is better than the current one but, is already in the tabu list, then it will be dismissed. Tabu list can also be called as short term memory, since it only ”blocks” solutions for a short period of time. Besides the short term memory, tabu search has also an intermediate memory and a long term memory, which make the search even more efficient.

- Intermediate memory - contains rules of intensification that allow to perform guided search in a direction where an optimum solution might be. Intensification rules perform only small changes in current solution in order to avoid deviation from the current search direction.

- Long term memory - contain rules of diversification that allow the algorithm to explore new search directions. Diversification rules are a set of methods that can change the current solution in a drastic way and thus explore new search directions. Normally, diversification rules are used when tabu search enters the stagnation phase.

Tabu search is a widely studied method and was largely applied in vehicle routing optimization problems. Results obtained by (Bräysy and Gendreau, 2002) show that tabu search is capable to find good solutions in short periods of time.

Algorithm 6: Tabu search.

input : $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ and set of UAVs $\mathcal{V}$.
output: Best solution found

```
while stopping criteria is not satisfied do
    Generate neighbourhood
    Select best neighbour
    Update current solution
    Update tabu memory
end
```

3.6.3 Simulated Annealing

Simulated annealing is a set of procedures inspired on thermodynamic laws that describe atom's behaviour at different temperature levels (Lin et al., 2011). When the temperature is high the atoms can move more easily, in routing optimization this state corresponds to a stage when it is possible to make drastic changes in the current solution. A lower temperature reduces the atoms movements (Figure 3.10). In optimizations problems this correspond to a neighbourhood exploration, i.e., only small changes are made in the current solution.

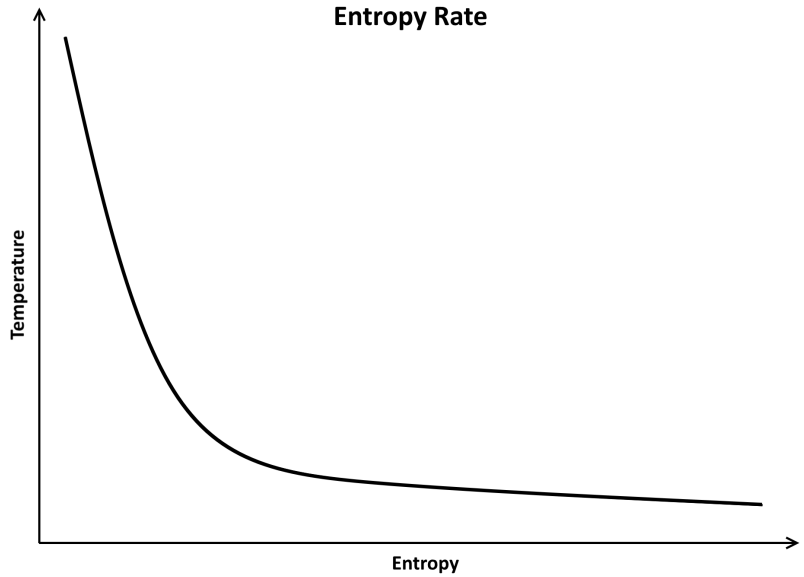


Figure 3.10: Entropy rate.

The main limitation of this method is the fact that it is difficult to discover when the temperature must be reduced. Despite that, results obtained by Chiang and Russell (1996) in VRP are very promising.

Algorithm 7: Simulated annealing.

input : $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ and set of UAVs $\mathcal{V}$.
output: Best solution found

```
while temperature != lower bound do
    while number of tries for current temperature != 0 do
        Randomize according to the current temperature
        if new solution is better that current solution then
            Replace current solution with new solution
        end
    end
    Decrease temperature by specified rate
end
```

3.6.4 Multiple Ant Colony Systems (MACS)

This algorithm is based on the behaviour of ant colonies, where ants cooperate to achieve a certain goal. In multi ant colony systems (MACS), each colony tries to minimize a single objective of the original problem (Gambardella et al., 1999). For instance, in vehicle routing problems, one colony minimizes the number of mobile elements required and the other one minimizes the travelling time. However, optimizing each objective separately does not solve the original problem. This requires a cooperation between the ant colonies which is done by pheromones exchange that are left by each colony.

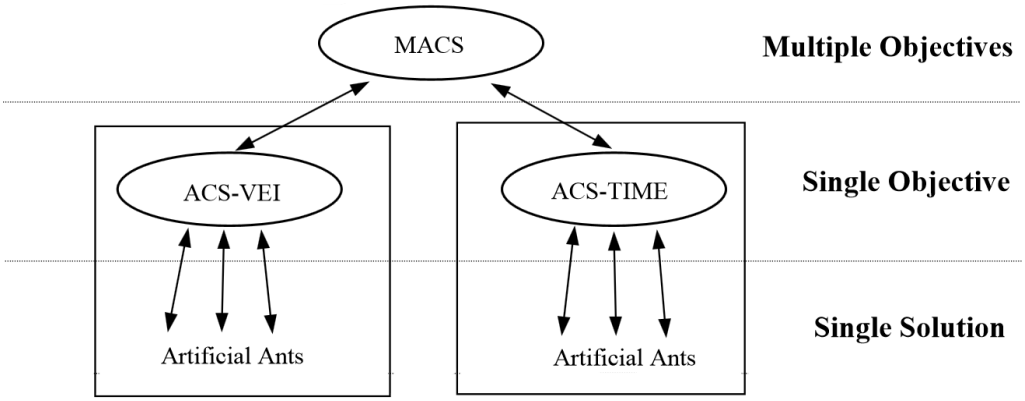


Figure 3.11: Multiple ant colony systems Gambardella et al. (1999).

Research by Gambardella et al. (1999) has a more detailed description about MACS and its methods.

3.6.5 Guided Local Search

This method (Mills et al., 2003) is quite similar to the Local Search (Chapter 3.5.2). The main difference is that Guided Local Search can penalize the current solution in order to escape local optimums. When the algorithm enters into a stagnation phase, i.e., is trapped in local optimum, an augmented cost function is used. The idea behind the augmented cost function is to penalize neighbourhood solutions and consequently make all neighbourhood less attractive than other solutions located in more distant neighbourhoods and thus, deviate the search direction to other search spaces. The augmented cost function can be expressed in the following way:

$$h(s) = g(s) + \lambda \times \sum_{i=1}^{n} P_{f_i} \times I_{f_i} \tag{3.6}$$

Where:

- s current solution
- g original cost function
- $\{ f_1, ..., f_n \}$ is a set of attributes that a solution possess
- Binary indication function I_{f_i}, which determines if a certain attribute is present in a solution.
- Penalization value P_{f_i}, which penalizes solutions that have attribute f_i. Initially set to 0 but is adapted with time.

The parameter λ can be adapted to allow diversification or intensification of the search. Higher the value of λ more diverse will be the search, inversely lower the value of λ more directional and guided will be the search. Research performed by Kilby et al. (1997) show that this method is capable to find near optimum solutions in multiple cases of Solomon benchmark (Solomon, 1987).

CHAPTER 4

Data Gathering in WSNs

4.1 Introduction

In previous sections the main challenges of the data gathering process have been stated. The mathematical formulation of the problem tackled in this thesis was introduced and some of general solving methods have been explored. In this section, the tackled problem will be reformulated, to incorporate the particularities of using mobile elements in WSNs, and a new hybrid heuristic approach to design paths for mobile elements in WSN will be presented and discussed.

4.2 Extended Problem

Given a set of sensor nodes that collect data about surrounding environment and a set o UAVs that will gather the data, the goal is to design an efficient set of paths to gather the data without violating the problem constraints.

Each sensor node has it own coordinates that identify its position. Moreover, each node has different buffer size to store the information and different sampling rate. This means that each node will fill its own buffer at different times. If the buffer size and the sampling are known then, it is possible to estimate the time window in which the buffer will be almost full and when it will be necessary to transfer the data to another entity, in this case the UAV. The transfer process changes according to the buffer size and throughput. By knowing these two informations it is possible to find out the time required to transfer the data from the sensor node to a UAV. To avoid losing the stored data, each node must be visited by an UAV within its own time window. Once the data is transferred to the UAV it must be delivered to the base station, for further processing, within a certain deadline. The deadline can be seen as a Time To Live (TTL), and in this case it is a time label. The data must be delivered to a base station within that TTL. However, if TTL is large the UAV can visit and gather the data from other nodes. The TTL restriction, which must be added to the initially discussed mathematical formalization of the data gathering problem (see Chapter 2.4), can be expressed mathematically as shown in expression 4.1.

$$\sum_{j \in N \setminus \{0\}: a_{jv} \geq a_{iv}} \sum_{h \in N \setminus \{0\}} t_{jh} x_{jhv} \leq DL_i, \qquad \forall i \in C, \forall v \in V \tag{4.1}$$

However, the constraint 4.1 can not be given as input to Mixed Integer Linear Programming (MILP) optimizers like CPLEX (IMB CPLEX, 2015) or Gurobi (Gurobi, 2015) as these solvers does not support the summation indexes that include variables. However, this limitation could be overcome by rewriting the constrain 4.1 as follows.

$$k_{iv}^i = dl_i \times \sum_{j \in N \setminus \{0\}} x_{ijv}, \qquad \forall i \in C, \forall v \in V \tag{4.2}$$

$$k_{iv}^l - s_j - t_{ij} + K(1 - x_{ijv}) \geq k_{jv}^l, \quad \forall l \in C, \forall v \in V, \forall i \in C, \forall j \in N \setminus \{0\} \tag{4.3}$$

$$k_{iv}^i \geq 0, x_{ijv} \in \{0, 1\}, \qquad \forall i, j \in N, \forall v \in V \tag{4.4}$$

where:

Sets	$V = \{1, 2, ..., v\}$	set of UAVs
	$N = \{0, 1, ..., n, n+1\}$	set of nodes including the base station
	$C = \{1, 2, ..., n\}$	set of nodes
Constants	dl_i	delivery limit of node i
	t_{ij}	travel time between node i and j
	s_i	service time at node i
	K	is a large value
Decision variables	x_{ijv}	one if UAV $v \in V$ travels from node i to node j, $i, j \in N$
	k_{jv}^i	remaining delivery limit time if vehicle $v \in V$ is carrying data belonging to node $i \in C$

The expanded constraints 4.2 and 4.3,originated from 4.1, ensure that the delivery limit value, of data collected at any node $i \in C$, decreases until the base station is reached. This decrease takes the travel and service time into consideration. Since all variables are non negative, data will not arrive outdated to the base station.

Overall, the problem to be solved is a Data Gathering Problem (DGP) applied to WSNs. In this case in particular, such DGP can be stated as a Vehicle Routing Problem with Time Windows (VRPTW), described in Chapter 2.4, with extra constraints so that:

data collected from a specific node reaches the base station within a certain time, called delivery limit and denoted by dl_{n_i}. That is, data has an expiration, or time-to-live (TTL), label meaning that it can not arrive outdated at the base station. Vehicle routing and scheduling problems are known to be NP-hard, see (Lenstra and Kan, 1981), meaning that the DGP will also be hard to solve within an acceptable period of time.

4.2.1 Input and Output

Now that the problem has been clearly defined, it is possible to identify what will be the input and the output for the purposed algorithm. The input information must contain:

- A set of sensor nodes. Where each node has:
  - geographical coordinates
  - buffer size
  - time windows - start and end time
  - delivery limit (TTL)
- A set of UAVs characterized by its:
  - Travelling speed
  - Buffer size
- A graph with information about the edges linking the sensor nodes
- Throughput during data transfer

With these input parameters, the algorithm must minimize the distance and the number of paths required to gather the data from all sensor nodes. The output must provide detailed information about the obtained solution:

- Number of paths
- Total distance
- Distance of each route
- Duration of each path
- Sequence of visited nodes
- Arrival time to each sensor node
- Departure time from each node

4.3 Proposed Hybrid Heuristic Algorithm

4.3.1 Problem Limits

Since this thesis tackles the combinatorial optimization problem it becomes necessary to establish the limits of the solution. Finding the optimum solution requires solving the mathematical problem with exact methods, which as seen in Chapter 2 is already a difficult task by itself. However, establishing an upper bound of the problem is more accessible. Usually, to set the upper bound constructive heuristics are used, because they are fast and, if well designed, can produce good results. In this thesis a customized Push Forward Insertion Heuristic (Chapter 3.4.4) was used to set the upper bound.

4.3.2 Adapted Push Forward Insertion Heuristic

The classical PFIH cost function was described in Chapter 3.4.4 but this cost function does not take into account the delivery limit constant. Hence, in order to have a more precise and more suitable way to evaluate each node, some modifications had to be done. The extension involves the addition of two parameters, the difference between end (et_i) and start (st_i) times of each node and the delivery limit (dl_i) of each one. The λ and τ are additional weight parameters. Larger values of λ will make more preferable nodes with smaller time windows to be inserted first in the solution. In the same way, larger values of τ will make preferable nodes with small delivery limit. Despite the increase of the number of weight parameters their sum is always equal to 1. The expression 4.5 represents the extended cost function.

$$c_i = -\alpha \times d_{i,0} + \beta \times et_i + \gamma((p_i/360) \times d_{i,0}) + \lambda(et_i - st_i) + \tau \times dl_i \tag{4.5}$$

A series of tests were performed to search the best combination for α, β, λ, γ and τ values and the best combination found looks as follows:

- $\alpha = 0.4$;
- $\beta = 0.2$;
- $\lambda = 0.1$;
- $\gamma = 0.1$;
- $\tau = 0.2$.

All simulations and the results shown in the following sections use these values as these provided, in most of cases, the best results for the DGP under consideration.

The fact of PFIH being a greedy heuristic combined to the way the algorithm starts (empty solution) produces, in many occasions, solutions far from optimum which offer a lot of space for improvements. One way to improve the performance of the constructive algorithms is to create an initial partial solution, done as discussed next.

4.3.3 Seeded Partial Solution

Since the PFIH is capable of inserting nodes in any position of the routes, provided feasibility, it is possible to create a partial solution *a priori*, i.e., before the execution of the algorithm. However, the creation of the partial solution should not be randomly generated (Cardoso et al., 2015).

Given the buffer size of the nodes, $\sum_{i \in C} d_i$, and the UAVs buffer size, q_v it is possible to estimate the minimum required number of paths to solve the problem by making a simple math division. This estimation of the number of paths is a lower bound since time restrictions and possible temporal conflicts are not taken into account. The PFIH algorithm will latter take those conflicts into account and solve them by starting a new path or paths, if necessary. An excessive and unnecessary number of initial paths can reduce the quality of the solution. Thus, the use of such estimation of initial paths ensures the quality of the final solution.

Once the number of initial paths is computed it is necessary to choose a set of nodes with as many elements as the number of initial paths. Then nodes from such set are inserted into the initial paths, one node per path (*BS, node, BS*). This way, at this stage, each UAV will only gather the data from a single node. The choice of nodes is based on their geographical location. This process is summarized in Algorithm 8.

Algorithm 8: Seeded partial solution.

input : Set of nodes (*C*), depot and set of UAVs (*V*)
output: Feasible partial solution

$AvgBuffSizeNode \longleftarrow$ Calculate average node buffer size
$AvgBuffSizeUAV \longleftarrow$ Calculate average UAV buffer size
$NumInitialRoutes \longleftarrow (AvgBuffSizeNode \times numSeeds) \div AvgBuffSizeUAV$
$SeedList \longleftarrow$ add BS
for $i \leftarrow 1$ **to** *NumInitialRoutes* **do**
    $FarthestNode \longleftarrow$ find farthest node from all nodes in $SeedList$
    $SeedList \longleftarrow$ add $FarthestNode$
end

The process described in Algorithm 8 allows to choose a set of well dispersed nodes that will be used to create a partial solution. Once the partial solution is built, the customized PFIH will receive the partial solution, the list of remaining nodes and the graph as input. Despite these changes, the PFIH algorithm functionality is maintained. In its next

step the PFIH evaluates each of remaining nodes and inserts each one in a path where the insertion causes the minimum increment of distance and the feasibility is satisfied. Figure 4.1 sketches the partial solution produced by the method described (right side) that was created for the problem presented at the left side.

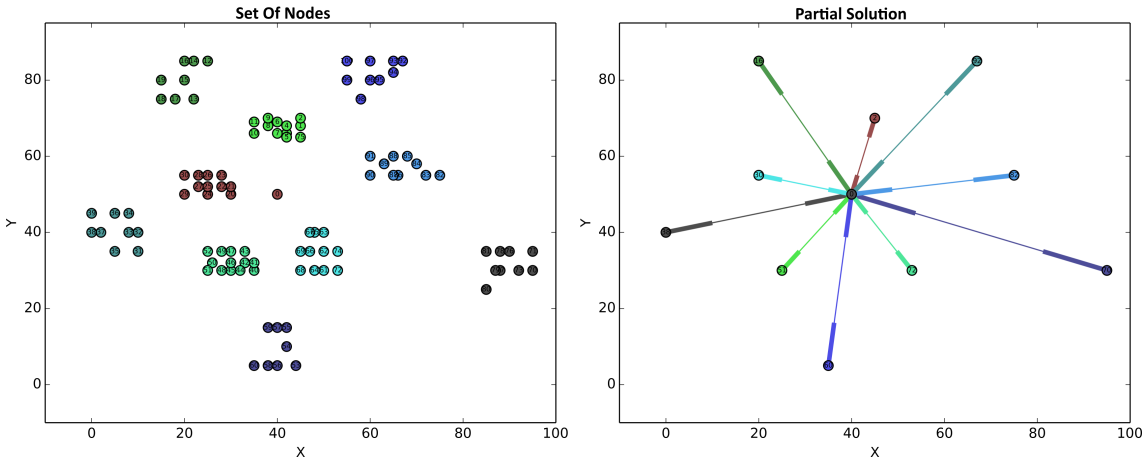


Figure 4.1: Seeded partial solution.

A simple test set was developed to study the performance of the PFIH algorithm with and without a partial initial solution. The obtained results have shown that the creation of a partial solution, especially in clustered cases as shown in Figure 4.1, by the Algorithm 8 and the subsequent use of it in PFIH algorithm improves considerably the quality of the solution.

4.3.4 Improvement Methods

Once the extended PFIH algorithm is executed a feasible solution is produced that in most cases, can be optimized by improvement methods. The methods that were used to improve the solution will be discussed in the following sections.

Intra Route Operator

An intra route operator, as the name suggests, performs operations on a single path (Chapter 3.5.2). One of the most commonly used operators from this set of methods is the 2-Opt operator. This operator goes through all paths, one by one, and rearranges the sequence by which the nodes are visited in order to reduce the distance of the path. Figure 4.2 could represent a solution, produced by PFIH described previously, for a hypothetical problem. The base stations represent the start and the end points of each path. At each path, nodes are visited according to the sequences shown in Figure 4.2. Thereby, to gather the data of 11 nodes three paths were necessary. The first one has a length of 35.86 units, the second one of 23.78 and the third of 48.61. The total distance distance is 108.25 units.

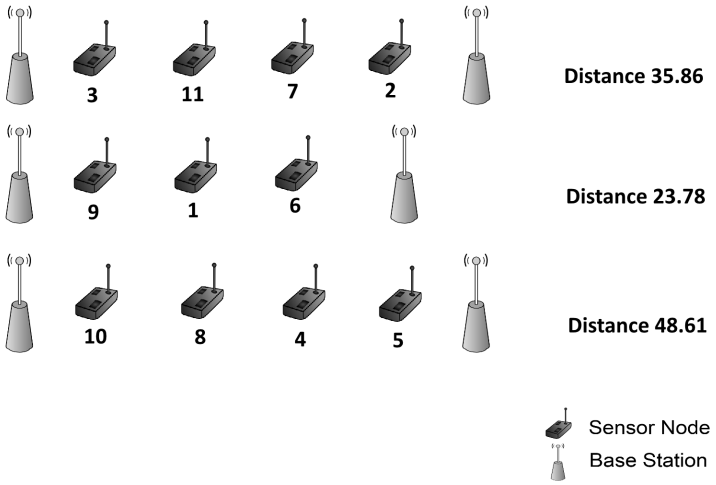


Figure 4.2: Solution representation

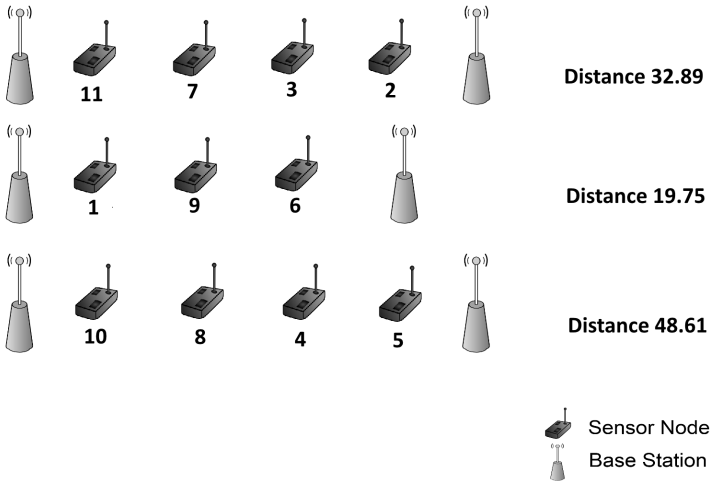


Figure 4.3: 2-Opt operator.

To illustrate the use of 2-Opt, let use see Figure 4.3 that could represent a solution produced by 2-Opt when applied to a solution from Figure 4.2. As one can see, the sequence by which the nodes are visited have suffered changes. The sequence of the first path changed from $BS \rightarrow 3 \rightarrow 11 \rightarrow 7 \rightarrow 2 \rightarrow BS$ to $BS \rightarrow 11 \rightarrow 7 \rightarrow 3 \rightarrow 2 \rightarrow BS$, leading to a distance reduction from from 35.86 to 32.89 units. The second path only suffered a small change, node 9 swapped with node 1. However, this slight change has reduced the distance of the second path in 4.03 units. The third path did not suffer any changes, i.e., any possible alteration would increase the distance of the path. After the execution of 2-Opt the total distance decreased from 108.25 to 101.25 units.

Inter Route Operator

Inter route operator performs a set of operations over a pair of routes in order to reduce the total distance of the solution. These methods reallocate a node, or a set of nodes

from one path to another in order to optimize the current solution. The implementation of this method was inspired by the genetic algorithms (Chapter 3.6.1), more precisely on a crossover operator called One Point Crossover (Magalhães-Mendes, 2013). This method receives two paths as input, for a later node reallocation, and returns two new paths. This operator is shown in Figure 4.4. The input paths are $BS \rightarrow 4 \rightarrow 8 \rightarrow 1 \rightarrow 5 \rightarrow 3 \rightarrow BS$ and $BS \rightarrow 7 \rightarrow 2 \rightarrow 6 \rightarrow 9 \rightarrow BS$. The crossover point is located at the third position. The output, two new routes, is $BS \rightarrow 4 \rightarrow 8 \rightarrow 6 \rightarrow 9 \rightarrow BS$ and $BS \rightarrow 7 \rightarrow 2 \rightarrow 1 \rightarrow 5 \rightarrow 3 \rightarrow BS$

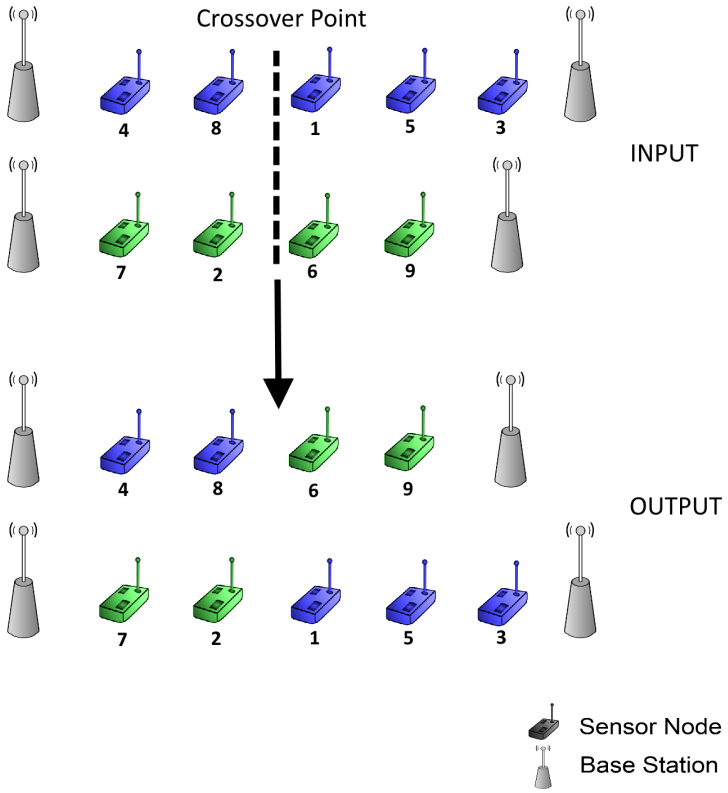


Figure 4.4: Crossover operator.

In summary, the local search operators (Chapter 3.5.2) are responsible for path optimization, i.e., distance reduction. They are not capable to reduce the number of paths. In order to reduce the number of path other methods should be used, as discussed next.

4.3.5 Node Ejection

In order to reduce the number of paths it is necessary to eject a set of nodes and reinsert them in a solution without violating the constraints of the problem. But how to eject nodes? Eject all nodes from a certain path and then reinsert them? If so, how to choose the right path? Randomly? It could work in some cases but usually in order to insert a node or a set of nodes in an existing path it is necessary to reallocate some of the nodes from the current path to other paths. And how about randomly eject a certain number of

nodes from the solution? In this case, there is no guarantee that this approach will work properly. Normally, completely random approaches are not a good fit for a combinatorial optimization problems.

In order to avoid random approaches the band neighbourhood ejection, inspired on the ruin and recreate principle described in Chapter 3.5.1, method was developed, which will be discussed in next.

4.3.6 Band Neighbourhood Ejection

Band neighbourhood ejection is a generalization method of the radial ejection presented in (Schrimpf et al., 2000). The methods select a path and for each node located in it, ejects a certain number of neighbour nodes. The ejection is based on the proximity and similarity of the nodes.

Algorithm 9: Band neighbourhood ejection.

input : Feasible solution
output: List of ejected nodes and a partial solution

```
1  Create roulette
2  Run roulette
3  Eject nodes from the chosen route
4  Set the number of neighbours to eject per node of the chosen route
5  foreach ejected node do
6      for i ← 1 to NumNeighToEject do
7          Find a similar neighbour
8          Eject it from the solution
9      end
10 end
```

In more detail, the first step will be to choose a path where the ejection will start. The path is chosen by a roulette method, inspired by the rank selection of genetic algorithm selection methods (Jebari and Madiafi, 2013). To build a roulette it is necessary to sort the paths of the solution in an ascending order, from the shortest to the longest one. Next step is to give each path its own ”slice” in the roulette. We consider that shortest paths (less customers) are defective so they will have a bigger ”slice” in roulette. The probabilities of ejection for the *i*-worst route, previously sorted by the number of nodes, is calculated according to expression 4.6 where *n* is the number of paths.

$$\frac{2i}{n(n+1)} \tag{4.6}$$

Once the route is chosen and the nodes to be ejected are known it becomes necessary to find the neighbours from other paths that will also be ejected.

Removing nodes from more distant neighbourhoods with completely different time restrictions, becomes inefficient since the probability of a future reinsertion and the consequent reduction of the number of paths and distance is considerably low. This means that only similar neighbours should be ejected. A similar neighbour is a node that is located in the proximity of the current node, also with similar time restrictions.

Figure 4.5 shows the band neighbourhood ejection procedure on a hypothetical example. The zoomed region is where the band ejection took place. After this procedure 5 nodes were ejected from the solution.

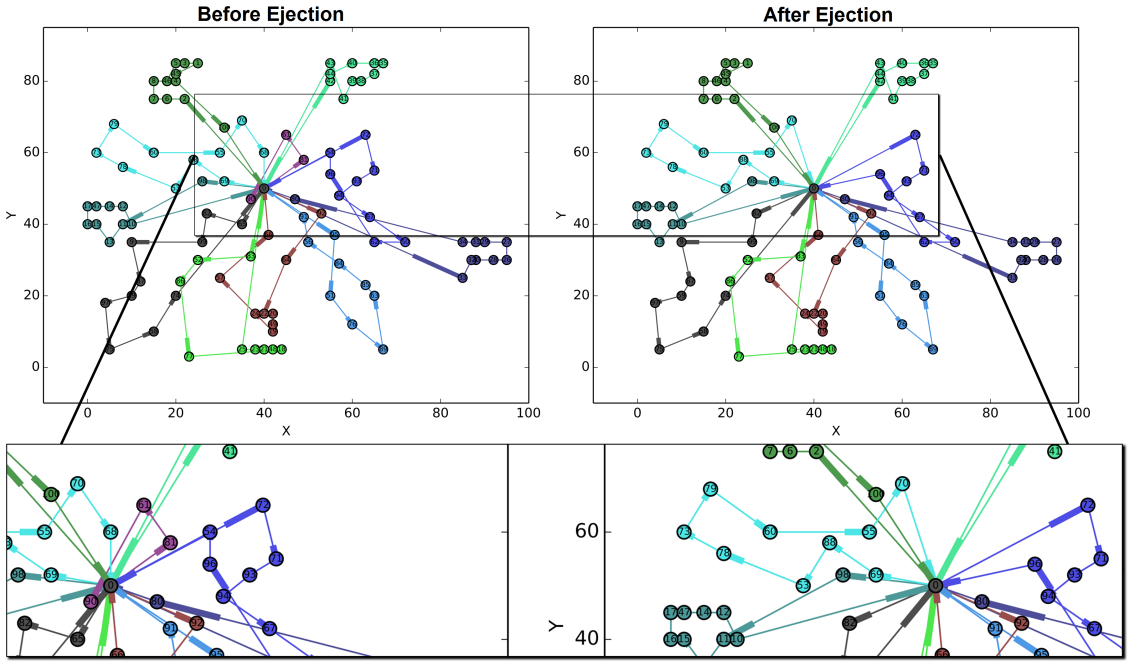


Figure 4.5: Band neighbourhood ejection.

Algorithm 9 provides a more detail description of this procedure.

4.3.7 Overall Procedure

Now that the core of the algorithm has been described it is time to put all pieces together and see how the algorithm really works. First let us take a look at Algorithm 10 and then discuss its particularities.

Lines 1 to 3 are the initialization steps of PFIH algorithm. The first step (Line 1) generates an evenly spaced partial solution. The second step (Line 2) is when the weights of cost function are defined. This step establishes the sequence by which the sensor nodes will be inserted in the solution. Once this is done, the PFIH is ready to go. In Line 3, the PFIH algorithm is executed and returns a feasible solution. The returned solution is stored in a variable called $CurrentSolution$. The remaining instructions of the algorithm are designed to improve the $CurrentSolution$, but in order not to loose the progress, a

copy of $CurrentSolution$ is stored in a variable called $BackUpSolution$ (Line 4). The variable $BestSolution$ (Line 5) stores the best solution found by the algorithm.

In Line 6 the tabu list (see Chapter 3.6.2) is initialized and the $CurrentSolution$ is inserted into this list. The goal of this list is to avoid unnecessary exploration of already explored search directions. By not accepting the already discovered solutions the search direction is guided into new areas of the search space. The not acceptance of already explored solutions is assured in Line 17. However, after a certain number of iterations, the solutions located in the tabu list will be acceptable again (Line 18). This way we also ensure that the algorithm is not guided in a completely opposite direction, away from the optimum solution. The $varNoSucces$ (Line 7) is a control variable that counts the number of iterations without any improvement.

In this algorithm the stopping criteria is the elapsed time $timeElapsed$ (Line 8), although this is not mandatory and it can be easily changed to any other stopping criteria. The variable $threshold$ (Line 9) plays an important role in the performance of the algorithm. The $threshold$ sets a limit on how worse the distance of $CurrentSolution$ can be when compared with $BackUpSolution$. The threshold variable can not be set too high because it would misguide the search. However, a well chosen $threshold$ can be very useful. The acceptance of a slightly worse solution, allows the algorithm to escape from local optimums without deviating from the search direction.

The remaining instructions of the algorithm works in the following way: a double loop is executed (Lines 11 and 12) until the stopping criteria is satisfied (Line 10). The double loop was inspired on the simulated annealing (see Chapter 3.6.3) and means that for each $ejectionRate$, which sets the number of neighbours to eject from the solution, the algorithm can perform a certain number of tries $varNoSucces$ to improve the current solution. However, contrary to the simulated annealing, each time the solution is improved we consider that the search is in a right direction and the $varNoSucces$ is reset. In Line 13 a set of nodes is ejected, accordingly to $ejectionRate$, and stored in the variable called $ejectedNodesList$. Next, in Line 14, the nodes are reinserted with PFIH algorithm in the $CurrentSolution$. The improvement methods are applied on the newly formed solution. The first one, 2Opt, is a classical intra route operator (Line 15) and the other one, one point crossover, is an inter route operator (Line 16). The condition in Line 17 ensures that we are not searching for a solution in an already explored search space. The remaining conditions, from Line 19 until Line 38, assess the quality of a newly formed solution and accordingly to the assessment discard or store the solution.

Algorithm 10: Routing algorithm.

input : $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ and set of UAVs $\mathcal{V}$.
output: Final solution

```
1  SeededSolution ⟵ Generate seeded partial solution
2  Define values of α , β , λ , γ , τ for PFIH and sort nodes by their PFIH cost
3  CurrentSolution ⟵ RunPFIH(SeededSolution)
4  BackUpSolution ⟵ CurrentSolution
5  BestSolution ⟵ CurrentSolution
6  tabuList ⟵ CurrentSolution
7  varNoSucces ⟵ 0
8  timeElapsed ⟵ currentTime
9  threshold ⟵ Define value of threshold
10 while timeElapsed < timeLimit do
11     for ejectionRate ← lowerLimit to upperLimit do
12         while varNoSucces < varLimit do
13             ejectedNodesList ⟵ BandNeighbourhoodEjection(ejectionRate)
14             CurrentSolution ⟵ PFIH(CurrentSolution, ejectedNodesList)
15             Apply 2-Opt to CurrentSolution
16             Apply Crossover to CurrentSolution
17             if CurrentSolution does not exists in tabuList then
18                 Update tabuList
19                 if CurrentSolution number of paths ≤ BackUpSolution number of paths
                   then
20                     if CurrentSolution distance ≤ BackUpSolution distance ×threshold
                       then
21                         BackUpSolution ⟵ CurrentSolution
22                         varNoSucces ⟵ 0
23                     else
24                         CurrentSolution ⟵ BackUpSolution
25                         increment varNoSucces
26                     end
27
28                     if CurrentSolution distance ≤ BestSolution distance then
29                         BestSolution ⟵ CurrentSolution
30                         varNoSucces ⟵ 0
31                     end
32                 else
33                     CurrentSolution ⟵ BestSolution
34                     increment varNoSucces
35                 end
36
37             else
38                 increment varNoSucces
39             end
40
41         end
42     end
43 end
```

CHAPTER 5

Performance evaluation

5.1 Introduction

In previous sections the problem tackled in this project has been presented and some real life applications have also been introduced. Beside that, a mathematical formalization of the problem and heuristic approach have been presented. In this chapter we will take a look at the data sets used to measure the performance of the heuristic algorithm. For performance measure either the optimal or the best known results for the VRPTW are used as reference. We also draw some conclusions and highlight some of our findings about the algorithm.

5.2 Data Set and Best Known results

Due to the relatively recent use of mobile elements for data gathering in WSNs there is no standardized data set to be used meaning that there is no effective way to compare performance between different algorithms. The most common approach in the sensor network community is to generate a large number of topologies and report average results. However, we believe that it is important to use data sets from the scientific community, even if adapted to DGP under analysis.

5.2.1 Generating Data Sets

To evaluate the presented algorithm the set of VRPTW instances, from Solomon's benchmark (Solomon, 2015b), have been used. Namely, instances with 100 nodes in the form of random (R), clustered (C) and random-clustered (RC) geometric distributions. The VRPTW instances provide, for each node, the coordinates, demand, time window and service time. For the vehicles, in this case UAVs, there is also information about their capacity. There are 6 sets of instances. Sets R1 and R2 have randomly generated geographical data, sets C1 and C2 have clustered geographical data, and sets RC1 and RC2 have a mix of randomly and clustered geographical data. Problem sets R1, C1 and RC1

have a short scheduling horizon while sets R2, C2 and RC2 have a long scheduling horizon, allowing more sensor nodes to be serviced by the same UAV. At each set, the vehicle (UAV in our case) capacity and service time does not change between instances.

These instances, however, do not include the delivery limit of nodes, dl_{n_i}. For a network scenario (instance) we define a range of delivery limits as follows. The lowest delivery limit should take into consideration the time required to return to the depot, i.e., $dl^{MIN} = max_{n_i \in N}\{d_{(n_0,n_i)}\}$. Also, using a delivery limit that exceeds the end time value of the base station would reduce the problem to the classic VRP. This value determines the arriving deadline to the base station, which must be accomplished, meaning that delivery limits higher than this value will have no effect on the solution, as it would not restrict the problem. That is, $dl^{MAX} = et_{n_{|\mathcal{N}|-1}}$. Therefore, the delivery limits for a network scenario (instance) should take values from dl^{MIN} to dl^{MAX}, $dl^{GAP} = dl^{MAX} - dl^{MIN}$ being the gap between these bounds. To evaluate different slack levels for the delivery limit at nodes the following six ranges have been used:

$$\Delta_1 = [dl^{MIN}, dl^{MIN} + 20\% dl^{GAP}]$$
$$\Delta_2 = [dl^{MIN} + 20\% dl^{GAP}, dl^{MIN} + 40\% dl^{GAP}]$$
$$\Delta_3 = [dl^{MIN} + 40\% dl^{GAP}, dl^{MIN} + 60\% dl^{GAP}]$$
$$\Delta_4 = [dl^{MIN} + 60\% dl^{GAP}, dl^{MIN} + 80\% dl^{GAP}]$$
$$\Delta_5 = [dl^{MIN} + 80\% dl^{GAP}, dl^{MIN} + 100\% dl^{GAP}]$$
$$\Delta_6 = \infty$$

These ranges have been applied to every network scenario (instance) under analysis. For a specific scenario the delivery limit of a node will be a random number at the range under consideration. This means that each instance of the original VRPTW problem will give rise to six instances of the data gathering problem under study, each with a different delivery limit range.

The Δ_6 is a special case of delivery limit. When the delivery limit is set to infinite the problem becomes a classic VRPTW. Although the main goal of this thesis is to solve the problem with the delivery limit, it is important to see the behaviour of the algorithm in all cases, even when the delivery limit is infinite (Δ_6).

5.2.2 Best Known Results

Best known results for VRPTW, used in the following section for comparison, have been extracted from Sintef web site (VRPTW, 2015), which maintains an updated list of the best known results and the list of methods used to obtain them. Currently there is no known algorithm that is capable to find the best solution for all the instances of Solomon's

benchmark. At the time this thesis is being written the best known list is composed of 19 different algorithms.

5.3 Getting the Results

5.3.1 Heuristic Results

The proposed heuristic has been applied 25 times to each generated instance, and for all data sets. The results shown in the following sections are an average performance of these executions. Tests were performed on a commodity computer (Core i7-4770 with 16GB of RAM). In all cases, the execution time, per instance generated, of the proposed heuristic did not exceed 10 seconds.

5.3.2 Exact Results

Our initial approach to obtain the exact results was to try to solve the original mathematical formalization previously described using the CPLEX package. However, we noticed that for data sets with just 25 nodes and 25 vehicles the mathematical representation contained over a half of million of restrictions. In case of data sets with 100 nodes the number of restrictions exceeded several millions. The enormous amount of restrictions combined with the complexity of the problem made impossible for CPLEX to solve it and find the optimum solution.

To reduce the number of restrictions and to somehow facilitate the search for the optimum solution we first established the upper bound number of UAVs with the proposed heuristic. This way, the solution returned by the proposed algorithm gave us the maximum number of UAVs required to solve the problem. If the proposed algorithm, which may find suboptimal solutions, uses a certain number of drones then CPLEX, which always finds the optimal solution, would never use more drones than the proposed heuristic. This way, instead of having all data sets with 25 UAVs, each instance had its own number of mobile elements accordingly to the solution obtained by the heuristic.

This approach proved to be correct because it allowed us to reduce the number of restrictions for problems with 25 nodes to a 100 thousand instead of a half a million. However, even after this procedure CPLEX was not able to solve all of the benchmark's instances.

When nodes are distributed randomly, as happens in sets R1, R2, RC1, and RC2, CPLEX is not able to find the optimal solution due to RAM limitations, in this case it was 64GB. Same happens for the C1 and C2 sets when the delivery limit is tight, namely between Δ_1 and Δ_2 and some other isolated cases with wider delivery limits.

This means that we can only compare the performance of the algorithm in clustered cases (C1 and C2) with delivery limit higher than Δ_2.

A distributed version of CPLEX was used to obtain the optimal results. All tests were performed on two commodity computers (Core i7-4770 with 16GB of RAM) and on a server (Xeon E5-2690 with 32GB of RAM) giving in total 32 logical processors and 64GB of RAM.

5.4 Analysis of the Results

5.4.1 Optimal vs Heuristic Solutions for Sets C1 and C2 with 25 Nodes

Tables 5.1 and 5.2 show the optimal and heuristic values obtained for clustered instances with Δ_3, Δ_4, Δ_5 and Δ_6, respectively. The coloured rows highlight the instances where there is more similarity between heuristic and optimal values throughout different values of Δ.

For Δ_3 CPLEX was unable to find optimal solutions for three instances, i.e., the results are non available (N/A). In these cases CPLEX exceeded the 64GB of memory with an average of 35% of search space still to explore, which means that the solutions found in the remaining 65% could be sub-optimal and they were not used in the comparison. As for the performance of the proposed heuristic we can notice that for Δ_3 it requires in average more 21% of UAVs. Although this might sound an high value we highight that an increase from 2 to 3 UAVs leads to an increase of 50%. For example, solutions found by the heuristic for instances C101, C102, C105, C106 and C107 require an additional UAV to visit all the nodes. As for the instance C108 the minor difference in the number of paths indicates that most of the 25 solutions generated by the heuristic have the same number of UAVs as the optimal solution. For the remaining 5 instances, that are comparable, the heuristic found solutions with the same number of paths/UAVs. However, in this cases the heuristic could not find the optimal solution as the distance variation it this set of problems is, in average, located in the range of 8%. In summary, for delivery limit equal to Δ_3, we can state that the algorithm need more tuning in order to obtain better results. However the heuristic has the advantage of finding solution much faster.

For Δ_4, delivery limit range between 60%-80%, the CPLEX was capable to obtain solutions for all instance except one (C104). It is also visible that as the delivery limit gets wider the distance required to visit all of the nodes decreases. For 9 of these instance the heuristic was able to use the same number of UAVs as the optimal solutions, and for two of these instances, namely C108 and C109, the solutions found were actually optimal in terms of distance. When compared with Δ_3, we can notice that the algorithm performs better in minimizing the number of UAVs, as the number of instances with

optimal number of UAVs increased from 5 to 9. In average, for this set, the proposed method requires 0.5 or 20% more UAVs and 7% more distance to visit all the nodes.

Table 5.1: Heuristic's performance for delivery limit equal to Δ_3 and Δ_4

DeliveryLimit				Δ_3				Δ_4			
	Time Windows			CPLEX		Heuristic		CPLEX		Heuristic	
Name	Min.	Avg.	Max.	P	D	P	D	P	D	P	D
C101	37	60	89	4	249.20	5	265.65	3	198.60	5	213.10
C102	43	325	1135	3	216.35	4	234.34	3	193.92	3.92	211.82
C103	43	588	1136	N/A	N/A	4	273.62	3	192.10	3	194.59
C104	43	852	1136	N/A	N/A	4	261.58	N/A	N/A	3	192.23
C105	75	121	117	4	249.20	5	261.19	3	195.14	3	206.63
C106	29	156	387	4	249.20	5	272.58	3	195.14	5	202.30
C107	189	189	189	4	223.08	5	236.97	3	202.87	3.04	221.64
C108	149	243	353	4	259.62	4.16	279.33	3	195.14	3	195.15
C109	360	360	360	N/A	N/A	4	261.85	3	191.81	3	191.81
C201	160	160	160	2	228.19	3	247.29	2	215.54	3	237.29
C202	160	937	3289	2	220.52	2	236.00	2	215.54	2	230.82
C203	160	1714	3290	2	220.52	2	236.00	2	215.54	2	231.54
C204	160	2492	3291	2	215.34	2	233.29	2	215.34	2	233.29
C205	320	320	320	2	225.66	2	241.59	2	215.54	2	234.94
C206	299	486	707	2	215.54	3	235.05	2	215.54	3	235.05
C207	177	612	1253	2	221.09	3	239.12	2	215.34	3	231.56
C208	640	640	640	2	215.37	2	236.15	2	215.37	2	234.94
Average				2.79	229.21	3.48	250.09	2.50	205.53	3.00	217.57

Coloured: higher similarity between heuristic and optimal values throughout different Δ values.

In case of delivery limit within Δ_5 range, we can observe that the algorithm performs quite well. In all comparable instances the number of UAVs required by heuristic is equal to optimal and in 11 cases it was able to find optimal solutions. In average the algorithm requires around 1% more distance when compared with the optimum.

Finally, when delivery limit is in Δ_6 range, when the problem comes down to a classic VRP, the proposed algorithm performs almost flawlessly. The heuristic found optimal solutions for all except the C202 and C208 instances, where the distance difference is 3% and 1% respectively.

Taking everything into account and analysing the highlighted rows it is possible to state that the heuristic performs better under the following conditions:

- Low/medium maximum time window sizes, but only when the minimum and average time window sizes approach the maximum time window size (C108);
- Very high maximum time window sizes (C202, C203, C204);
- When the maximum, average and minimum time window sizes are equal, but only for the highest values (C205,C208).

In general, considering the performance of the algorithm with different Δ ranges we might get an indication that different strategies could exist for different types of sets. That is, sets with highly variable time windows could render λ the dominant factor, while for

Table 5.2: Heuristic's performance for delivery limit equal to Δ_5 and Δ_6

DeliveryLimit				Δ_5				Δ_6			
	Time Windows			CPLEX		Heuristic		Optimal (Solomon, 2015a)		Heuristic	
Name	Min.	Avg.	Max.	P	D	P	D	P	D	P	D
C101	37	60	89	3	191.81	3	191.81	3	191.30	3	191.81
C102	43	325	1135	3	190.74	3	191.81	3	190.30	3	190.74
C103	43	588	1136	3	190.74	3	190.74	3	190.30	3	190.74
C104	43	852	1136	N/A	N/A	3	188.53	3	186.90	3	187.45
C105	75	121	117	3	191.81	3	191.81	3	191.30	3	191.81
C106	29	156	387	3	191.81	3	191.81	3	191.30	3	191.81
C107	189	189	189	3	191.81	3	191.81	3	191.30	3	191.81
C108	149	243	353	3	191.81	3	191.81	3	191.30	3	191.81
C109	360	360	360	3	191.81	3	191.81	3	191.30	3	191.81
C201	160	160	160	2	215.54	2	228.96	2	214.70	2	215.54
C202	160	937	3289	2	215.54	2	220.52	2	214.70	2	220.52
C203	160	1714	3290	2	220.26	2	220.26	2	214.70	2	215.54
C204	160	2492	3291	2	213.93	2	213.93	2	213.10	2	213.93
C205	320	320	320	2	215.54	2	215.54	2	214.70	2	215.54
C206	299	486	707	2	215.54	2	218.90	2	214.70	2	215.54
C207	177	612	1253	2	215.34	2	215.54	2	214.50	2	215.54
C208	640	640	640	2	215.37	2	229.43	2	214.50	2	215.57
Average				2.50	203.71	2.53	205.00	2.53	201.82	2.53	202.80

Coloured: higher similarity between heuristic and optimal values throughout different Δ values.

the remaining sets other factors would dominate. That is, α, β, γ, λ and τ should dynamically follow the intrinsic nature of the sets and additional operators might be necessary to find optimal solutions for all considered instances.

The complete table of results can be seen in Appendix.

5.4.2 Heuristic Solutions for all Sets with 25 Nodes

The following plots relate to the total distance travelled by the UAVs, and the total number of paths/UAVs required, obtained by the heuristic approach for all sets. Sets C1, C2, R1, R2, RC1 and RC2 include between 8 and 12 instances and, although the capacity of UAVs and service time is the same for all instances of a set, the demand, time window and delivery limit of nodes will be different from instance to instance. This means that there might be different performances for different instances. For this reason, the solution's distance and number of UAVs of instance i undergoes the following formulas so that the increasing factor (IF) of each instance is obtained:

$$IF_i^{Distance} = \frac{Distance_i - Distance_i^*}{Distance_i^*} \tag{5.1}$$

$$IF_i^{NumUAV} = \frac{NumUAV_i - NumUAV_i^*}{NumUAV_i^*} \tag{5.2}$$

where $Distance_i$ and $NumUAV_i$ is the distance and number of UAVs obtained by the proposed hybrid heuristic and $Distance_i^*$ and $NumUAV_i^*$ are the best known solutions identified by heuristics for the VRPTW, obtained from Solomon (2015b). The best known results from VRPTW have been used because no DGP best known results are known, and

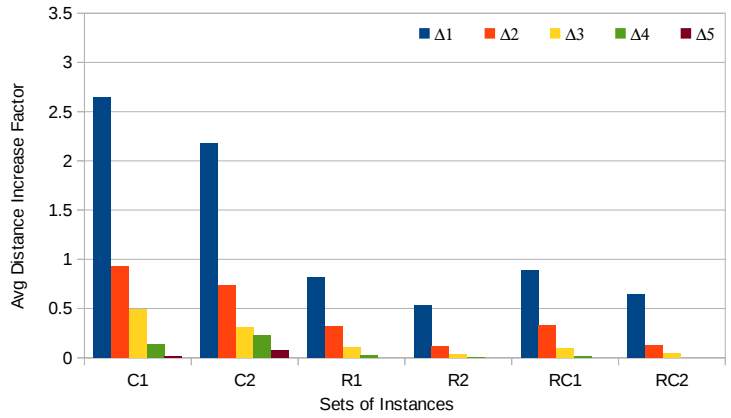


Figure 5.1: Average distance increase factor.

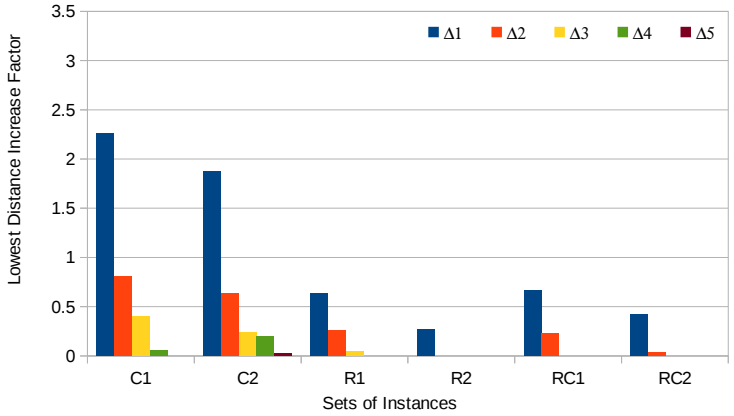


Figure 5.2: Lowest distance increase factor.

also because DGP gets similar to VRPTW as the delivery limit approaches infinity.

Figure 5.1 relates to distances and shows the average distance increase factor, considering all instances of a set, for all sets under analysis and different delivery limit ranges. Figures 5.2 and 5.3 show the lowest and highest distance increase factors from all instances of a set, for all sets under analysis and different delivery limit ranges. These can be seen as the lower and upper bounds of algorithm's performance. From these three plots it is possible to observe that, as the delivery limit range becomes less tight the more the solutions obtained by the hybrid heuristic closely approximate the best known solutions for the VRPTW. Therefore, we can state that the proposed hybrid heuristic algorithm is expected to be a good approach for the DGP. The clustered sets C1 and C2 are the ones for which the hybrid heuristic performs worse. Therefore, this algorithm is more suitable when nodes are dispersed.

Concerning the lowest and highest distance increase factors, at Figures 5.2 and 5.3, the smallest the difference between them the higher the certainty degree for a specific set. That is, if a small difference is obtained for a specific set then the outcome of the

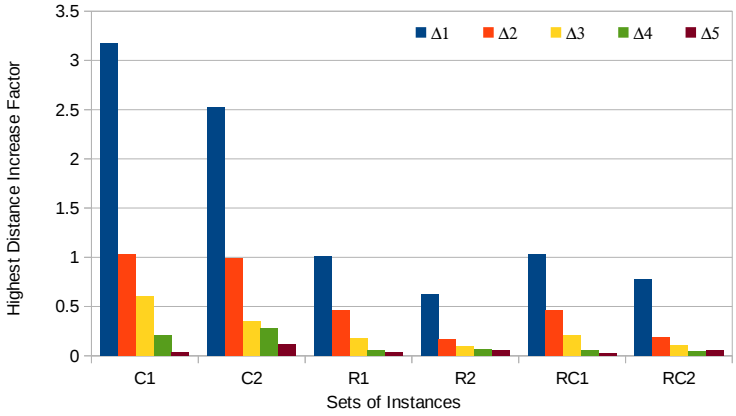


Figure 5.3: Highest distance increase factor.

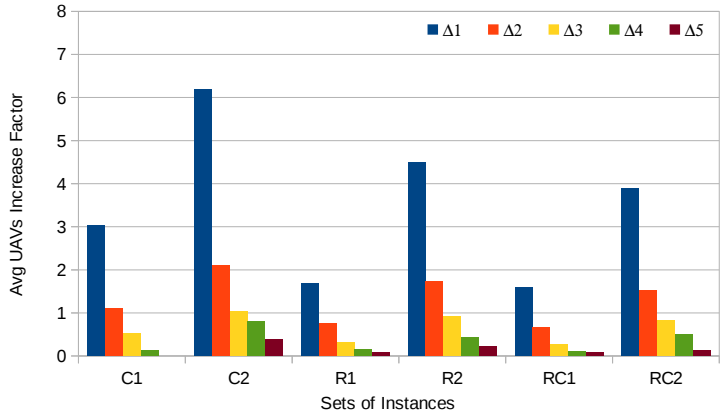


Figure 5.4: Average number of UAVs increase factor.

algorithm is more predictable whatever the network scenario (instance), fitting into that set, is used. Therefore, although the clustered sets perform worse, its performance is quite good, with a high certainty degree, for many ranges (Δ_2 to Δ_5). The certainty degree is also high for the other sets.

Figures 5.4, 5.5 and 5.6 also relate to average, lowest and highest increase factors, but now concerning the number of UAVs. We can observe that sets R2, C2 and RC2 with a long scheduling horizon, allowing more sensor nodes to be serviced by the same UAV, are the ones with an higher increase factor. This is so because delivery limits override the long scheduling horizon remove, avoiding UAVs to serve many sensor nodes. Therefore, the gap between DGP and VRPTW solutions will be higher. However, for ranges Δ_2 to Δ_5 the number of UAVs reduces significantly, also with high certainty degree. Since with sets R1, C1 and RC1, with short scheduling horizon, the hybrid heuristic algorithm closely approaches the best known solutions for the VRPTW, we can reconfirm that the proposed hybrid heuristic algorithm is expected to be a good approach for the DGP.

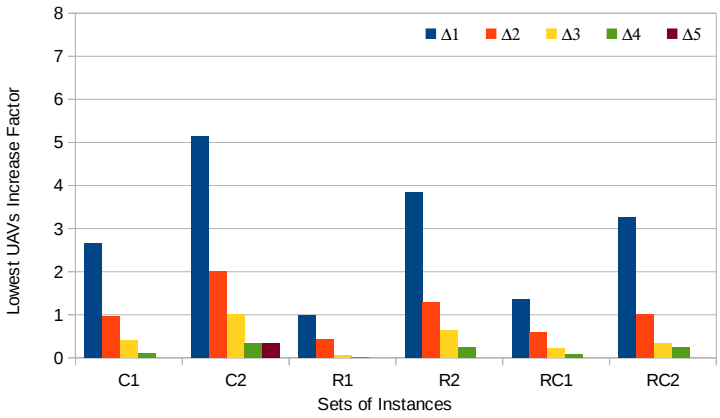


Figure 5.5: Lowest number of UAVs increase factor.

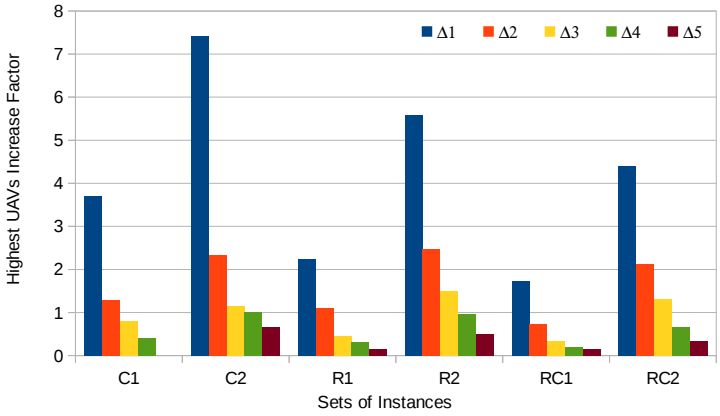


Figure 5.6: Highest number of UAVs increase factor.

5.4.3 Best known vs Heuristic results with 100 Nodes

Since, due to technical restrictions, it is impossible to study the performance of the algorithm in problems with more than 25 nodes with delivery limit. Let us take a look at the performance of the algorithm in larger problems without the delivery limit, namely at classic VRP problem with 100 nodes.

Table 5.3 summarizes the performance of the algorithm for each instance of Solomon's benchmark. As one can notice, the algorithm performs quite well in clustered cases (C1 and C2). The number of paths is equal to the best known solutions and the distance difference is negligible.

As for random instances (R1 and R2) it is noticeable that the proposed algorithm performs better in distance minimization than in minimization of the number of paths. In the majority of random instances our algorithm was able to get better distance than the best know algorithms. In R211 instance, by having an additional vehicle the proposed algorithm reduced the distance by an already considerable 9%.

For the mixed case, random and clustered instances (RC1 and RC2), the tendency remains. At the expense of an additional path the algorithm is able approach the best known distance and, in some cases, even improve it.

Considering all instances and their best known solutions obtained by 19 different algorithms, in average the proposed algorithm requires 6.5% more UAVs to produce solutions with the same distance as best known results.

Table 5.3: Results obtained with the proposed method for instances with infinite delivery limit of the Solomon's benchmark with 100 nodes.

	Best Found (VRPTW, 2015)		Heuristic		Δ	
Instance	P	D	P	D	Δ P %	Δ D %
C101	10	828.94	10	828.94	0.0%	0.0%
C102	10	828.94	10	828.94	0.0%	0.0%
C103	10	828.06	10	828.07	0.0%	0.0%
C104	10	824.78	10	827.82	0.0%	0.4%
C105	10	828.94	10	828.94	0.0%	0.0%
C106	10	828.94	10	828.94	0.0%	0.0%
C107	10	828.94	10	828.94	0.0%	0.0%
C108	10	828.94	10	830.33	0.0%	0.2%
C109	10	828.94	10	828.94	0.0%	0.0%
C201	3	591.56	3	591.56	0.0%	0.0%
C202	3	591.56	3	591.56	0.0%	0.0%
C203	3	591.17	3	591.17	0.0%	0.0%
C204	3	590.60	3	595.44	0.0%	0.8%
C205	3	588.88	3	588.88	0.0%	0.0%
C206	3	588.49	3	592.48	0.0%	0.7%
C207	3	588.29	3	588.29	0.0%	0.0%
C208	3	588.32	3	588.43	0.0%	0.0%
R101	19	1650.80	19	1680.91	0.0%	1.8%
R102	17	1486.12	17.84	1495.55	4.9%	0.6%
R103	13	1292.68	14	1223.96	7.7%	-5.3%
R104	9	1007.31	10	1000.05	11.1%	-0.7%
R105	14	1377.11	14.76	1402.19	5.4%	1.8%
R106	12	1252.03	12.8	1273.16	6.7%	1.7%
R107	10	1104.66	11	1082.28	10.0%	-2.0%
R108	9	960.88	10	992.43	11.1%	3.3%
R109	11	1194.73	12	1178.74	9.1%	-1.3%
R110	10	1118.84	11.44	1112.70	14.4%	-0.5%
R111	10	1096.72	11	1081.36	10.0%	-1.4%
R112	9	982.14	10	988.12	11.1%	0.6%
R201	4	1252.37	4	1296.11	0.0%	3.5%
R202	3	1191.70	4	1099.59	33.3%	-7.7%
R203	3	939.50	3	947.28	0.0%	0.8%
R204	2	825.52	3	762.00	50.0%	-7.7%
R205	3	994.42	3	1051.33	0.0%	5.7%
R206	3	906.14	3	937.19	0.0%	3.4%
R207	2	890.61	3	869.22	50.0%	-2.4%
R208	2	726.82	2.92	723.35	46.0%	-0.5%
R209	3	909.16	3	925.63	0.0%	1.8%
R210	3	939.37	3	985.94	0.0%	5.0%
R211	2	885.71	3	805.18	50.0%	-9.1%
RC101	14	1696.94	15.16	1676.87	8.3%	-1.2%
RC102	12	1554.75	13.72	1495.30	14.3%	-3.8%
RC103	11	1261.67	11	1291.23	0.0%	2.3%
RC104	10	1135.48	10	1195.32	0.0%	5.3%
RC105	13	1629.44	14	1626.79	7.7%	-0.2%
RC106	11	1424.73	12.48	1411.87	13.5%	-0.9%
RC107	11	1230.48	11.96	1244.28	8.7%	1.1%
RC108	10	1139.82	11	1149.98	10.0%	0.9%
RC201	4	1406.94	5	1360.35	25.0%	-3.3%
RC202	3	1365.65	4	1233.88	33.3%	-9.6%
RC203	3	1049.62	3	1134.94	0.0%	8.1%
RC204	3	798.46	3	833.90	0.0%	4.4%
RC205	4	1297.65	4	1353.76	0.0%	4.3%
RC206	3	1146.32	4	1104.13	33.3%	-3.7%
RC207	3	1061.14	3.4	1071.50	13.3%	1.0%
RC208	3	828.14	3	885.77	0.0%	7.0%
Average	7.23	1021.19	7.71	1020.92	6.5%	0.0%

CHAPTER 6

Conclusion and Future Work

6.1 Conclusion

This thesis addressed some of the existing issues of a generic data gathering problem within the WSN. As shown, classic multi hop approach may be useful due to its ability to rapidly deliver data to the desired destination. However, potential environment disasters can damage the communication channels and thus prevent the delivery of data. As shown, unmanned aerial vehicles (UAVs) can overcome these limitations as they do not require any additional infrastructure to gather the data. Some real life applications of UAVs in WSNs have been introduced and discussed in Chapter 1.

Two mathematical formalizations have been provided, one (in Chapter 2 for a generic data gathering problem, which may be used in a large set of similar problems, and an extended one (in Chapter 4) that describes the problem tackled in this thesis.

A deep analysis of the main classes of algorithms have been presented and some of the main approaches to solve routing problems have been presented in Chapter 3.

A custom hybrid heuristic capable to solve the DGP have been presented and its particularities have been described and discussed. Results obtained show that the proposed algorithm is capable to solve the DGP problem with an acceptable performance and, besides that, the algorithm is capable to achieve great results when compared with the classic VRP benchmark.

I hope that the research done in this thesis adds something new, no matter how small, to the state of the art algorithms used in the optimization area and that it enhances our knowledge on the data gathering problems.

6.2 Future Work

In this thesis a single base station routing problem was considered. Future work will focus on making the algorithm more generic and capable to deal with other routing problem variations such as multiple depots and dynamic nodes that can appear during the execution of the algorithm. Also the current algorithm can be more tuned to improve the results

obtained and other solving methods can be incorporated in it. In addition, the algorithm can include the communication range of the sensor nodes and thus allow to reduce the distance travelled by the mobile element. Another interesting feature that can be incorporated is the self managing node clusters where one of the nodes act as collector of the data. This would avoid the need to visit all nodes.

Ultimately the possibilities are endless.

Bibliography

Almi'ani, K., Viglas, A., and Libman, L. (2010). Mobile element path planning for time-constrained data gathering in wireless sensor networks. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 843–850. IEEE.

Baker, E. K. (1983). Technical note—an exact algorithm for the time-constrained traveling salesman problem. *Operations Research*, 31(5):938–940.

Bhadauria, D. and Isler, V. (2009). Data gathering tours for mobile robots. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3868–3873. IEEE.

Bräysy, O. and Gendreau, M. (2002). Tabu search heuristics for the vehicle routing problem with time windows. *Top*, 10(2):211–237.

Bychkovskiy, V., Megerian, S., Estrin, D., and Potkonjak, M. (2003). A collaborative approach to in-place sensor calibration. In *Information Processing in Sensor Networks*, pages 301–316. Springer.

Cardoso, P., Schütz, G., Mazayev, A., and Ey, E. (2015). Solutions in under 10 seconds for vehicle routing problems with time windows using commodity computers. In Gaspar-Cunha, A., Henggeler Antunes, C., and Coello, C. C., editors, *Evolutionary Multi-Criterion Optimization*, volume 9019 of *Lecture Notes in Computer Science*, pages 418–432. Springer International Publishing.

Carić, T., Galić, A., Fosin, J., Gold, H., and Reinholz, A. (2008). A modelling and optimization framework for real-world vehicle routing problems. *Vehicle Routing Problem*, 15.

Chiang, W.-C. and Russell, R. A. (1996). Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63(1):3–27.

Chin, J.-C., Rao, N. S., Yau, D. K., Shankar, M., Yang, Y., Hou, J. C., Srivathsan, S., and Iyengar, S. (2010). Identification of low-level point radioactive sources using a sensor network. *ACM Transactions on Sensor Networks (TOSN)*, 7(3):21.

Costa, F. G., Ueyama, J., Braun, T., Pessin, G., Osório, F. S., and Vargas, P. A. (2012). The use of unmanned aerial vehicles and wireless sensor network in agricultural applications. In *Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International*, pages 5045–5048. IEEE.

Florêncio, J. C. P. F. (2011). Análise comparativa de algoritmos de otimização para o problema de roteamento de veículos. Trabalho de Graduação.

Gambardella, L. M., Taillard, É., and Agazzi, G. (1999). Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In *New ideas in optimization*. Citeseer.

Ghoseiri, K. and Ghannadpour, S. (2009). Hybrid genetic algorithm for vehicle routing and scheduling problem. *Journal of Applied Science*, 9(1):79–87.

Gurobi, O. (2015). Gurobi. http://www.gurobi.com/. Accessed 06/04/2015.

IMB CPLEX, I. I. C. O. S. (2015). Cplex. http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/. Accessed 06/04/2015.

Jaheruddin, S. (2010). *Friendly Interchange Heuristic for Vehicle Routing Problems with Time Windows*. PhD thesis, Tilburg University.

Jebari, K. and Madiafi, M. (2013). Selection methods for genetic algorithms. *International Journal of Emerging Sciences*, 3(4).

Kilby, P., Prosser, P., and Shaw, P. (1997). Guided local search for the vehicle routing problem.

Lenstra, J. K. and Kan, A. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227.

Lin, S.-W., Yu, V. F., and Lu, C.-C. (2011). A simulated annealing heuristic for the truck and trailer routing problem with time windows. *Expert Systems with Applications*, 38(12):15244–15252.

Magalhães-Mendes, J. (2013). A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem. *WSEAS Transactions on Computers*, 12(4):164–173.

Mamun, Q. (2012). A qualitative comparison of different logical topologies for wireless sensor networks. *Sensors*, 12(11):14887–14913.

Mills, P., Tsang, E., and Ford, J. (2003). Applying an extended guided local search to the quadratic assignment problem. *Annals of Operations Research*, 118(1-4):121–135.

Moura, A. (2008). A multi-objective genetic algorithm for the vehicle routing with time windows and loading problem. In *Intelligent Decision Support*, pages 187–201. Springer.

Ombuki, B., Ross, B. J., and Hanshar, F. (2006). Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence*, 24(1):4–5.

Pignaton de Freitas, E., Heimfarth, T., Netto, I. F., Lino, C. E., Pereira, C. E., Ferreira, A. M., Wagner, F. R., and Larsson, T. (2010). Uav relay network to support wsn connectivity. In *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2010 International Congress on*, pages 309–314. IEEE.

Richard Pon, Maxim A. Batalin, J. G. (2005). Networked infomechanical systems: A mobile embedded networked sensor platform. *IEEE*, page 6.

Rodrigues, J. J. and Neves, P. A. (2010). A survey on ip-based wireless sensor network solutions. *International Journal of Communication Systems*, 23(8):963–981.

Russell, R. A. (1995). Hybrid heuristics for the vehicle routing problem with time windows. *Transportation science*, 29(2):156–166.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., and Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265.

Solomon, M. M. (2015a). Optimal solutions for c1 and c2 problems. http://web.cba.neu.edu/ msolomon/c1c2solu.htm. Accessed 06/04/2015.

Solomon, M. M. (2015b). Vrptw benchmark problems. http://web.cba.neu.edu/ msolomon/problems.htm. Accessed 06/04/2015.

Somasundara, A. A., Ramamoorthy, A., and Srivastava, M. B. (2007). Mobile element scheduling with dynamic deadlines. *Mobile Computing, IEEE Transactions on*, 6(4):395–410.

Sujit, P., Lucani, D., and Sousa, J. (2013). Joint route planning for uav and sensor network for data retrieval. In *Systems Conference (SysCon), 2013 IEEE International*, pages 688–692. IEEE.

Tam, V. and Ma, K. (2008). An effective search framework combining meta-heuristics to solve the vehicle routing problems with time windows. *Vehicle Routing Problem*, page 35.

Tan, K., Lee, L., and Ou, K. (2001). Artificial intelligence heuristics in solving vehicle routing problems with time window constraints. *Engineering Applications of Artificial Intelligence*, 14(6):825–837.

Tekdas, O., Isler, V., Lim, J. H., and Terzis, A. (2009). Using mobile robots to harvest data from sensor fields. *IEEE Wireless Communications*, 16(1):22.

Thangiah, S. R. (1999). A hybrid genetic algorithms, simulated annealing and tabu search heuristic for vehicle routing problems with time windows. *Practical handbook of genetic algorithms*, 3:347–381.

Thangiah, S. R., Osman, I. H., and Sun, T. (1994). Hybrid genetic algorithm, simulated annealing and tabu search methods for vehicle routing problems with time windows. *Computer Science Department, Slippery Rock University, Technical Report SRU CpSc-TR-94-27*, 69.

Thangiah, S. R., Osman, I. H., Vinayagamoorthy, R., and Sun, T. (1993). Algorithms for the vehicle routing problems with time deadlines. *American Journal of Mathematical and Management Sciences*, 13(3-4):323–355.

VRPTW, S. (2015). Sintef. https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/100-customers/. Accessed 06/04/2015.

Xing, G., Wang, T., Xie, Z., and Jia, W. (2008). Rendezvous planning in wireless sensor networks with mobile elements. *Mobile Computing, IEEE Transactions on*, 7(12):1430–1443.

APPENDIX A

Appendix

[This page intentionally left blank]

Table A.1: Adapted Solomon instances with 25 nodes

Delivery Limit	Δ_1		Δ_2		Δ_3		Δ_4		Δ_5	
Instance	Paths	Distance	Paths	Distance	Paths	Distance	Paths	Distance	Paths	Distance
C101	14	689.80	7	351.16	5	265.65	5	213.10	3	191.81
C102	10	505.19	6	324.82	4	234.34	3.92	211.82	3	191.81
C103	10	532.27	5	306.68	4	273.62	3	194.59	3	190.74
C104	10	524.27	5	290.84	4	261.58	3	192.23	3	188.53
C105	10	505.61	7	351.25	5	261.19	3	206.63	3	191.81
C106	10	559.40	7	353.37	5	272.58	5	202.30	3	191.81
C107	11	603.87	6.48	342.85	5	236.97	3.04	221.64	3	191.81
C108	11	578.92	6	316.90	4.16	279.33	3	195.15	3	191.81
C109	9	456.68	6	315.12	4	261.85	3	191.81	3	191.81
C201	8	518.49	4	351.69	3	247.29	3	237.29	2	228.96
C202	9	558.28	4	293.18	2	236.00	2	230.82	2	220.52
C203	6	437.05	4	291.65	2	236.00	2	231.54	2	220.26
C204	6	414.80	3	269.76	2	233.29	2	233.29	2	213.93
C205	9	532.81	4	311.48	2	241.59	2	234.94	2	215.54
C206	7	457.52	4	314.47	3	235.05	3	235.05	2	218.90
C207	7	442.34	4	323.03	3	239.12	3	231.56	2	215.54
C208	6	458.18	3	278.56	2	236.15	2	234.94	2	229.43
R101	15	841.72	10	676.88	9	631.79	8	618.33	8	618.33
R102	11	691.10	8	630.69	7	559.71	7	554.96	7	548.11
R103	10	665.50	6	553.02	6	506.59	6	471.91	5	455.70
R104	9	633.28	6	520.24	5	468.09	4	428.38	4	417.96
R105	12	758.83	8	614.78	6	531.54	6	531.54	6	531.54
R106	10	681.98	7	577.50	5	503.64	5	470.17	5	466.48
R107	9	648.32	7	539.24	5	464.14	4	441.96	4	425.27
R108	10	647.92	6	506.51	4	435.05	4	413.47	4	398.30
R109	10	690.47	6	501.28	5	481.75	5	459.85	5	442.63
R110	9	622.79	6.04	510.10	5	455.12	5	445.18	4.84	445.33
R111	10	673.35	6	516.82	5	464.34	4	435.56	4	429.70
R112	8	599.01	6	474.31	4	423.22	4	394.55	4	394.10
R201	8	702.02	5.8	567.69	4.8	519.10	3	474.37	3	474.37
R202	8	687.44	4	508.85	3.32	481.09	3	437.16	2	457.74
R203	8	630.10	4	477.46	3.32	451.00	2	412.46	2	400.40
R204	5.24	519.60	3	390.29	3	375.73	2	371.51	2	356.09
R205	6	561.26	4	451.13	3	411.42	3	394.06	2	405.98
R206	6	522.91	3	444.54	2.84	396.22	2.64	377.44	2	378.18
R207	5	494.60	3	407.34	2.96	390.74	2	369.37	1.44	383.00
R208	5.52	488.94	3	378.37	2	350.78	1	347.82	1	329.33
R209	6	525.17	3.36	411.99	2	381.01	2	371.56	2	371.62
R210	6	599.51	4	464.08	3	423.42	2	412.52	2	410.60
R211	4	445.49	2	366.06	2	352.59	2	352.09	1.12	362.29
RC101	9	788.35	7	619.62	5	483.35	4.28	475.94	4.36	479.27
RC102	9	758.15	6	541.61	4	449.00	3	352.74	3	352.74
RC103	9	738.50	6	529.48	4	395.07	3	333.92	3	333.92
RC104	8	677.41	6	516.91	4	388.83	3	309.22	3	307.14
RC105	9	749.25	6	576.58	5	507.18	4	412.38	4	412.75
RC106	8	718.26	5.92	519.69	4	403.11	3	346.51	3	346.51
RC107	7	650.83	5	471.95	3.12	320.22	3	298.95	3	298.95
RC108	7	604.46	5	463.58	3	299.19	3	294.99	3	294.99
RC201	11.48	1017.82	5	525.90	4	519.02	3.92	413.70	3	361.24
RC202	6.96	674.38	4.96	468.75	3	399.86	3	338.87	2	376.12
RC203	6	632.79	3	465.23	2	391.48	2.52	342.32	2	356.35
RC204	5	526.88	2	406.53	2	351.78	1.08	330.82	1.88	315.71
RC205	8	707.41	4.84	514.65	3	418.13	3	346.56	2	386.15
RC206	6	579.85	3	453.18	3.08	391.37	3	325.10	2	344.93
RC207	5	492.40	3	406.53	3	305.45	2	308.57	2	308.57
RC208	4	366.49	2.16	292.66	2	272.51	2	269.57	1.76	280.42

Table A.2: Adapted Solomon instances with 50 nodes

Delivery Limit	Δ_1		Δ_2		Δ_3		Δ_4		Δ_5	
Instance	Paths	Distance	Paths	Distance	Paths	Distance	Paths	Distance	Paths	Distance
C101	23	1450.37	12	788.66	8	570.26	7	481.47	5	363.25
C102	20.04	1152.44	11	689.70	7.44	500.88	6	415.68	5	387.26
C103	19.12	1128.04	10	633.07	7.2	536.12	5	408.77	5	363.92
C104	18	1031.69	10	614.52	7	504.11	5	368.03	5	359.96
C105	19.72	1134.73	11	721.95	8	539.54	6	418.14	5	363.25
C106	20.48	1193.88	11	686.24	8	546.19	6	517.26	5	363.25
C107	20.4	1252.69	11	724.87	8	499.24	6	422.59	5	363.25
C108	19.32	1186.86	10.2	677.34	7.96	526.96	6	402.33	5	363.25
C109	18.2	1086.92	10	635.00	7	522.56	6	403.82	5	363.25
C201	14	1017.25	6	590.28	5	495.62	4	412.39	3	375.21
C202	12.96	1015.57	5	574.71	3	548.77	3	391.27	2	427.85
C203	10	938.92	5	535.97	4	419.11	2	499.89	2	418.64
C204	9	749.59	5	498.23	3	412.55	3	395.71	2	371.54
C205	12.04	926.58	5.68	583.58	4	501.51	3.96	411.05	2	441.06
C206	10	848.60	6	575.57	4	471.20	3	405.55	2	416.23
C207	10.92	850.53	5	558.70	4	418.74	3	433.82	2	442.00
C208	11.44	912.77	5.52	520.17	4	418.07	3	427.21	2	352.29
R101	25	1682.23	18	1291.82	13	1113.72	12	1051.79	12	1051.62
R102	21	1460.81	14.24	1167.41	11	969.99	10	932.87	10	925.40
R103	18	1324.06	11.84	947.05	9.04	864.61	8	810.80	8	784.56
R104	16	1182.24	10	822.09	8	733.75	6.12	663.56	6	632.21
R105	21.04	1472.02	14.04	1156.62	10.48	974.82	9	914.31	9	914.31
R106	18.96	1353.22	12.24	1034.95	9.44	891.80	8	803.24	7	882.01
R107	17.4	1269.58	11.44	913.89	8.64	789.70	7	736.59	6	744.08
R108	15	1081.49	10	821.42	7	687.60	6	649.40	6	621.85
R109	19	1359.12	12	985.48	9	828.36	8	805.81	7.92	801.58
R110	16.32	1233.08	10.28	892.32	8.04	758.65	7	713.20	7	708.29
R111	16	1175.30	11	896.76	8	761.39	7	719.01	7	709.53
R112	15	1098.42	9.4	812.58	8	698.69	6.32	668.39	6	640.59
R201	13	1271.26	6	987.27	5	882.38	3	896.04	3	863.96
R202	11.48	1177.17	5	854.96	4	783.71	3	774.28	2	816.78
R203	10	1052.57	4	845.08	3	770.65	2	755.98	2	663.60
R204	8	820.05	3.2	577.05	3	524.67	2	527.66	2	509.25
R205	10	1044.98	4.8	813.00	3	793.58	3	717.98	2	747.07
R206	8.92	955.09	4	771.91	3	693.91	2	668.94	2	658.74
R207	8	853.06	4	698.18	3	624.01	2	615.08	2	594.35
R208	7.72	785.95	3.56	565.54	2.08	541.43	2	507.97	2	494.17
R209	8.76	943.68	4.2	703.91	3	655.00	2	670.24	2	661.34
R210	10.16	995.89	5	762.58	3	711.39	2	724.91	2	675.41
R211	7	744.32	4	597.83	3	566.79	2	566.28	2	561.25
RC101	19	1753.05	12	1167.15	9	974.13	8	948.70	8	948.73
RC102	18	1675.47	11	1092.62	8	927.01	7	823.97	7	823.97
RC103	17	1544.14	10	975.97	8	845.43	6	713.13	6	712.56
RC104	14	1282.08	10	929.83	7	776.42	5	554.25	5	549.34
RC105	17	1524.64	11.92	1117.90	8	919.14	8	858.26	8	856.97
RC106	17	1587.57	10	962.93	8	876.92	7	803.09	6	724.65
RC107	16	1455.09	10	936.41	7	746.15	6	645.54	6	645.51
RC108	14.96	1326.85	9	872.41	7	777.72	6	599.17	6	599.17
RC201	14.68	1633.38	6	1057.78	4	1082.01	4	787.20	3	840.07
RC202	10.92	1288.03	5.88	933.36	4	888.89	3	784.82	2	885.72
RC203	9.12	1175.93	4.96	833.41	4	713.46	3	621.15	2	674.42
RC204	7	825.72	4	596.56	3	549.77	2	488.48	2	479.87
RC205	11.04	1377.30	6	990.51	4	927.34	3	819.33	3	761.38
RC206	9.92	1156.58	5	806.32	4	725.13	3	668.50	2	777.90
RC207	9.04	1048.38	5	740.01	3	637.80	3	583.93	2	655.81
RC208	7	712.69	3.76	534.54	3	513.43	2	534.48	2	494.89

Table A.3: Adapted Solomon instances with 75 nodes

Delivery Limit	Δ_1		Δ_2		Δ_3		Δ_4		Δ_5	
Instance	Paths	Distance	Paths	Distance	Paths	Distance	Paths	Distance	Paths	Distance
C101	34.16	2375.28	17.32	1194.55	12	903.16	10	828.34	8	688.04
C102	32.64	2126.47	16.16	1131.95	11	813.11	8	718.63	8	666.91
C103	30.04	1972.37	14.96	1046.99	11.04	892.73	8	686.36	8	666.06
C104	28	1843.50	15	1097.14	10.04	834.40	8.08	702.44	7	674.72
C105	29.88	2093.62	16.48	1143.01	11.6	895.44	9	712.83	8	650.35
C106	30.36	2059.37	16.16	1098.66	12.08	922.12	9.08	724.88	8	664.46
C107	29.84	2063.68	16.24	1172.01	11	829.22	9	709.90	8	662.27
C108	29.6	1988.45	15.68	1126.92	11	856.24	8.52	711.08	8	660.10
C109	27.36	1836.82	15.12	1096.38	10.96	861.28	8	686.18	8	640.80
C201	20.44	1419.58	8	771.43	6	659.16	6	635.72	3	545.28
C202	19	1662.92	8	770.97	5	656.78	4	624.27	3	520.64
C203	15.24	1366.17	7	801.49	5	642.44	4	593.66	3	516.01
C204	13.68	1184.08	7	737.93	4.92	614.07	4	562.30	3	514.20
C205	19.44	1388.64	8	817.14	6	664.38	5	597.87	3	519.88
C206	16.12	1421.55	7.84	818.02	6	651.78	4	612.21	3	510.71
C207	14.52	1256.14	7.28	855.17	5	732.73	4	644.78	3	529.78
C208	16.4	1325.24	7	787.00	5	662.51	4	620.90	3	519.50
R101	35	2399.67	24.44	1814.86	17.96	1549.43	16	1439.76	16	1436.66
R102	28.88	2104.93	20.28	1636.05	14.72	1376.69	14	1304.21	14	1296.17
R103	25	1849.85	16	1336.46	12.52	1151.10	11	1061.52	11	1039.59
R104	22.96	1688.11	14.24	1146.84	11	982.10	9.04	885.86	8	826.84
R105	29.12	2136.62	18.72	1530.76	13.72	1293.63	12	1181.15	12	1169.51
R106	25.92	1863.46	17.04	1402.22	12.56	1198.91	11	1096.69	10	1130.37
R107	23.68	1760.51	15	1251.30	11.6	1066.76	9.32	953.25	9	938.67
R108	21.28	1600.16	14.36	1185.28	10	944.61	8.6	861.71	8	812.84
R109	24	1819.61	15.32	1289.39	12	1106.51	10	1044.99	10.12	1031.44
R110	23	1678.86	14.76	1205.78	11	1036.81	9.8	961.66	9.24	956.71
R111	23	1715.58	14.96	1236.52	11	985.26	9	947.46	9	909.49
R112	22	1626.25	13	1100.68	10.44	966.48	9	876.55	8	844.40
R201	16.08	1697.89	7.96	1242.38	6	1164.24	4	1120.65	3	1152.01
R202	14.72	1540.65	7	1085.22	5	1094.60	4	995.85	3	1012.17
R203	14.6	1435.79	6.56	1004.82	4	963.38	3	901.43	3	831.99
R204	10.96	1069.78	5	754.65	4	688.70	3	677.26	2	739.87
R205	12.64	1374.08	6.28	1009.83	5	927.22	3	943.29	3	901.80
R206	12.04	1257.56	6	933.10	4	856.81	3	823.19	2	886.80
R207	11.04	1168.35	5.52	899.12	4	808.99	3	774.28	2	791.55
R208	10	998.83	4.92	717.24	3.6	665.37	2.08	648.11	2	628.51
R209	12.68	1255.15	5.96	864.80	4	795.67	3	808.76	3	797.09
R210	12.08	1307.32	6	964.97	4	895.62	3	877.80	3	846.20
R211	10	986.41	5	765.95	3.64	719.67	3	697.29	2	737.62
RC101	28	2459.94	18.44	1752.84	13.76	1478.96	12.16	1399.62	12.16	1394.89
RC102	26.04	2325.62	16.36	1599.64	12.96	1355.90	11	1280.14	11	1259.17
RC103	24.04	2126.48	15.6	1482.59	11	1215.08	10	1102.63	9.6	1092.62
RC104	21	1884.91	14.88	1404.59	10.44	1130.87	9	978.99	8	960.49
RC105	24.52	2166.54	17	1588.80	12.8	1340.66	11	1256.26	11	1260.18
RC106	25	2252.71	14.08	1411.18	11.6	1232.13	10	1141.39	10	1141.06
RC107	22.64	2049.30	13.8	1336.45	10	1098.03	9	1015.05	9	1013.16
RC108	20.68	1816.21	12.96	1222.80	10	1048.37	8.96	939.55	8.96	940.60
RC201	18.2	2091.13	8.36	1433.34	7	1268.36	5	1181.64	4	1178.23
RC202	14	1671.93	8	1242.96	5	1303.06	4	1117.57	3	1168.73
RC203	12.12	1509.94	6.24	1045.07	5	967.77	3	1004.67	3	907.49
RC204	10.48	1165.95	5	781.02	4	719.20	3	691.37	2	716.94
RC205	14.24	1788.46	7.8	1317.91	6	1160.50	4	1133.43	4	1086.07
RC206	12.88	1518.24	6.16	1093.83	5	953.38	4	894.81	3	948.44
RC207	11.92	1412.33	6	980.92	4.92	872.20	4	813.07	3	859.19
RC208	10	1079.78	5.32	786.23	3.96	711.93	3	694.78	2.96	696.00

Table A.4: Adapted Solomon instances with 100 nodes

Delivery Limit	Δ_1		Δ_2		Δ_3		Δ_4		Δ_5	
Instance	Paths	Distance	Paths	Distance	Paths	Distance	Paths	Distance	Paths	Distance
C101	47.08	3463.35	22.8	1657.09	18	1326.34	14	996.91	10	849.64
C102	42.6	3121.97	21.12	1599.80	14.84	1195.29	11	1002.29	10	854.68
C103	39.44	2878.63	19.72	1495.05	14.44	1253.59	11	916.45	10	844.07
C104	38.64	2829.74	20.04	1535.83	14	1152.26	11	904.92	10	834.08
C105	40.04	3164.93	21.68	1594.27	15.96	1263.00	11	950.83	10	828.94
C106	39.24	3004.81	20.88	1583.93	15.88	1304.47	11	982.30	10	852.76
C107	39.84	3078.72	21.72	1683.31	15	1207.30	11	921.79	10	844.39
C108	38.56	2906.63	20.92	1633.45	14.56	1236.96	11	909.92	10	844.74
C109	36.64	2706.23	20.2	1630.68	14.12	1214.72	11	878.07	10	828.94
C201	25.28	2081.75	9.16	964.22	6	797.92	6	710.20	5	662.28
C202	23.92	2068.95	9	1177.24	6	800.25	6	712.30	4	647.33
C203	19.68	1859.31	9.36	1006.44	6	734.66	5	719.81	4	634.37
C204	18.4	1696.39	9	1066.89	6	790.09	4	754.55	4	633.63
C205	24.44	1917.87	10	994.84	6.24	777.28	6	721.04	4	637.87
C206	20.68	1869.35	9.16	986.11	6.12	765.74	6	702.89	4	604.87
C207	19.2	1713.67	9.12	979.20	6	744.58	5	748.61	4	641.38
C208	21.08	1816.41	9.36	1011.24	6.44	774.29	5	723.33	4	634.04
R101	42.52	2828.45	27.2	2085.44	22.28	1795.68	19.12	1692.06	19	1668.62
R102	33.88	2428.67	24.52	1916.38	18	1557.85	18	1485.05	18	1481.16
R103	30.6	2166.97	21	1638.87	16.04	1398.02	14	1266.28	14	1224.80
R104	29.08	2023.20	18.8	1406.02	13	1170.00	11.76	1067.21	10	1003.89
R105	34.48	2474.71	22	1736.22	16.32	1498.30	15	1413.60	14.88	1391.16
R106	30.76	2209.09	20.08	1626.95	15.24	1400.36	13	1287.63	13	1261.18
R107	28.88	2073.33	18.88	1489.53	14.04	1243.14	11.64	1150.69	11	1082.59
R108	27.04	1902.98	18	1406.39	12.96	1129.36	10.88	1005.54	10	992.17
R109	30.04	2157.38	19.04	1536.45	14.08	1278.57	12.64	1195.68	12.04	1196.31
R110	28.52	2004.49	18	1413.11	14	1224.77	11.96	1121.38	11.48	1113.82
R111	29.08	2055.14	18	1431.01	13.12	1172.30	11.8	1108.76	11	1073.33
R112	27	1893.08	16.68	1311.73	13	1111.36	11	1028.09	10	986.28
R201	19.92	2039.68	9.24	1410.64	7	1338.59	5	1287.85	4	1256.83
R202	17	1698.22	8	1294.42	6	1214.22	4	1231.60	4	1112.26
R203	15.88	1505.56	7.96	1097.80	5	1030.62	4	1002.68	3	961.34
R204	12.72	1262.23	6.16	909.02	5	806.90	3	800.57	3	776.39
R205	15.16	1577.83	7.6	1136.15	5	1093.29	4	1016.09	3	1053.29
R206	14.6	1430.21	6.96	1035.71	5	979.24	4	927.16	3	940.56
R207	13.16	1299.29	6.96	991.46	5	902.26	3.92	865.70	3	838.48
R208	12.16	1160.38	6	830.33	4.28	781.73	3	739.93	3	722.30
R209	14.52	1424.34	6.88	991.65	4.88	913.18	3.96	897.66	3	947.07
R210	14.52	1497.62	7.4	1085.27	5	1009.11	4	964.50	3	960.68
R211	11.92	1136.39	6.52	854.73	4	797.62	3	801.57	3	790.70
RC101	33	2911.31	22.64	2169.93	17.04	1796.31	15.08	1674.31	15.4	1679.15
RC102	32.6	2858.28	20.2	1970.73	15.88	1633.19	14	1513.51	13.96	1502.84
RC103	29	2556.72	19.04	1827.01	14	1458.17	11.88	1299.62	11	1286.41
RC104	26.24	2309.99	17.36	1655.58	13.2	1374.72	11	1198.62	10	1164.60
RC105	31.88	2719.21	20.76	1998.15	16	1606.10	14.4	1612.90	14	1572.04
RC106	30	2667.60	18.4	1777.32	14.76	1514.94	13.04	1419.69	12.4	1416.36
RC107	27.96	2439.95	17.88	1667.33	13.32	1385.62	12	1266.81	11.92	1248.14
RC108	26.12	2246.16	16	1511.38	13	1290.53	11	1161.49	11	1149.13
RC201	20.88	2446.02	9.88	1604.33	7	1501.50	6	1418.19	5	1349.73
RC202	15.72	1943.40	9.36	1420.80	6.92	1310.61	5	1200.43	4	1186.46
RC203	16.2	1827.80	8.08	1199.87	6	1102.32	5	1013.03	3	1106.82
RC204	12.88	1420.52	7	934.24	5	881.12	4	832.90	3	831.96
RC205	18.24	2069.66	9.44	1537.86	7	1339.94	5	1295.02	4	1344.19
RC206	15.92	1854.87	8	1294.43	6	1186.89	4.92	1096.91	4	1089.63
RC207	14.16	1669.47	7.4	1176.31	5.32	1058.45	4.68	1012.81	3.48	1054.26
RC208	12.8	1373.58	6	878.41	4	895.28	3.92	810.35	3	873.84