UNIVERSIDADE FEDERAL FLUMINENSE

GUILHERME ROLIM E SOUZA

MODEL AND SOLUTION FOR THE DATA AGGREGATOR POSITIONING PROBLEM IN SMART GRIDS

NITERÓI

2016

UNIVERSIDADE FEDERAL FLUMINENSE

GUILHERME ROLIM E SOUZA

MODEL AND SOLUTION FOR THE DATA AGGREGATOR POSITIONING PROBLEM IN SMART GRIDS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Sistemas de Computação.

Orientador:
CÉLIO VINICIUS NEVES DE ALBUQUERQUE

Co-orientador:
IGOR MONTEIRO MORAES

NITERÓI

2016

GUILHERME ROLIM E SOUZA

MODEL AND SOLUTION FOR THE DATA AGGREGATOR POSITIONING PROBLEM IN SMART GRIDS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Sistemas de Computação.

Aprovada em maio de 2016.

BANCA EXAMINADORA

Prof. Célio V. N. de Albuquerque - Orientador, UFF

Prof. Igor Monteiro Moraes - Co-Orientador, UFF

Prof. Diego Gimenez Passos, UFF

Prof. Rodrigo de Souza Couto, UERJ

Dr. Arlan Luiz Bettiol, A Vero Domino

Niterói

2016

À minha família.

Acknowledgments

Aos meus pais, Gilma Caminha Rolim e Souza e Walder Santos e Souza, e meus avós, Guiomar Caminha Rolim e Gilson Rangel Rolim, por proporcionarem uma ótima base e incentivo para minha educação.

Ao meu irmão Felipe Rolim e Souza por me apoiar e me ajudar sempre que necessário.

Aos meus orientadores Célio Vinicius Neves de Albuquerque e Igor Monteiro Moraes pelas orientações que me permitiram evoluir e aperfeiçoar este trabalho.

Ao professor Diego Gimenez Passos, que me ajudou ao longo do desenvolvimento deste trabalho com sugestões e novas ideias.

Ao professor Raphael Pereira de Oliveira Guerra, que me deu uma ótima ideia de otimização para a implementação de um dos algoritmos deste trabalho.

À todos os meus amigos e companheiros de laboratório que me apoiaram. Em especial Mateus Carvalho Azis e João Felipe Nicolaci Pimentel que me ajudaram com ideias e dicas na implementação de uma ferramenta fruto deste trabalho.

À Marister Monteiro Luz do Outão, sempre disposta a resolver meus problemas.

Aos professores do Instituto de Computação da UFF, que compartilharam sua sabedoria e experiências ao longo desses anos.

À CELESC e ANEEL por financiarem o projeto TELE-SIRIS (SImulador de Redes InteligenteS) que atuou como base para o desenvolvimento deste trabalho.

À Universidade Federal Fluminense, à CAPES e à FAPERJ, que proporcionaram infraestrutura e recursos para a realização deste trabalho.

Resumo

As redes elétricas inteligentes (*Smart Grids*) representam a próxima etapa na distribuição eficiente de energia e exigem uma série de novas arquiteturas que se comunicam entre si. Esta nova abordagem implica que, em adição ao fluxo de energia, as redes elétricas inteligentes produzem um fluxo bidirecional de informação. A infraestrutura de medição avançada (*Advanced Metering Infrastructure* - AMI) é uma rede de comunicação de média distância cujo principal objetivo é prover comunicação entre medidores inteligentes, presentes em cada residência, e agregadores de dados (*Data Aggregation Points* - DAPs).

Este trabalho foca em determinar o número mínimo de DAPs, e suas posições, de forma que cada medidor inteligente seja capaz de transmitir seus dados através de um enlace bem estabelecido. Para tal, este problema é modelado para um problema de otimização conhecido como *Set Covering Problem* (SCP) que age sobre um conjunto de bons enlaces pré-calculados de acordo com estimativas de propagação de sinal. O modelo é estendido para considerar a técnica de múltiplos saltos e redundância de medidores, caso um DAP falhe.

Para solucionar o SCP, uma heurística baseada no método de divisão e conquista é proposta. A heurística divide o problema em subregiões de acordo com a estimativa do consumo de memória para solucioná-la. A divisão é realizada através do método de clusterização K-Means. Resultados mostram que a heurística proposta é capaz de encontrar soluções próximas à ótima para grandes instâncias, reduzindo drasticamente a quantidade de memória RAM necessária e o tempo de execução.

Palavras-chave: Redes Elétricas Inteligentes, Infraestrutura de Medição Avançada, Planejamento de Agregadores de Dados, Planejamento de Redes.

Abstract

Smart Grids represent the next step in efficient energy distribution and demand a series of new architectures that communicate with each other. This approach means that, in addition to energy, Smart Grids also induce a bidirectional information flow. The Advanced Metering Infrastructure is a middle-range communication network which main aim is to provide communication between smart meters, present in each home, and data aggregation points (DAPs).

This work focuses on determining the minimum number of DAPs and their positions in order that every smart meter is able to forward its data through a well established communication link. To do so, we model this problem as a well known optimization problem called Set Covering Problem that acts based upon a preprocessed set of good quality links determined by path loss propagation estimates. The model is further extended to consider multiple hops communication and smart meters coverage redundancy, in case a DAP fails.

To solve the SCP, a divide-and-conquer heuristic is proposed. The heuristic splits the problem in sub-regions according to a memory consumption estimate to solve it. The split is performed by the clustering method, K-Means. Results show that our proposed heuristic is capable of finding near-optimal solutions for huge instances while greatly reducing the required amount of RAM memory and execution time.

Keywords: Smart Grids, Advanced Metering Infrastructure, Data Aggregator Planning, Network Planning.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Current electrical grids are characterized by a unidirectional flow of electricity which is transmitted from generators to consumers. Based on a set of communication technologies, a new electrical grid architecture named Smart Grid is currently under development. Through an interconnected set of modules and infrastructures, Smart Grids are able to analyze energy demands in real time, allowing the optimization of energy generation, transmission and distribution [9] [31].

For Smart Grids to become possible, it is necessary that a smart meter is deployed in each residence. Smart meters allow remote charging [21] and are responsible for keeping track of their users' energy consumption which is transmitted to the power utility provider periodically. Nevertheless, smart meters must be integrated to the Smart Grid network using a communication technology. This integration is performed by the Advanced Metering Infrastructure (AMI), which provides coverage for smart meters through a set of Data Aggregation Points (DAPs). Data collected by these meters are periodically transmitted to one or more DAPs. The connectivity between DAPs and meters can be established either by wired or wireless technologies [26]. DAPs transmit the collected data from the neighborhood to its provider's processing center via long-distance communication technologies such as GPRS, 3G and 4G cellular network [20], LTE [4], or IEEE 802.16 (WiMAX) [12].

The most common wired communication technologies used between meters and DAPs comprehends the optical fiber and PLC (Power Line Communication). Optical fibers are a reliable and secure communication technology but possesses high cost of installation and maintenance [7]. PLC, on the other hand, takes advantage of the current electrical wiring to transmit data but is susceptible to faults, noises and interference generated by devices connected to the same power line [10]. Wireless technologies present an alternative

to reduce installation costs demanded by wired infrastructures. Moreover, the wireless infrastructure is disassociated from the electrical one, meaning that possible electrical failures (e.g. wire breaks) do not interfere with the communication between the AMI devices, assuming they are equipped with a battery or even solar panels. There is no standard on which communication technology should be used between meters and DAPs though short and medium distance communication standards such as IEEE 802.15.4 and IEEE 802.11 are recommended [12]. In this work it is assumed that the communication between meters and DAPs happens exclusively through wireless communication. By doing so, to correctly solve the studied problem, our proposed model takes into account the adversities imposed by wireless technologies.

One of the greatest challenges for an AMI is to properly plan the DAPs' positions. The manual analysis of the best DAPs' positions is costly and hard to execute in practice, especially in dense neighborhoods. Planning an entire city, for example, involves considering thousands of meters, one for each residence. This work focuses on determining the minimum number of DAPs and their positions such that every smart meter is able to forward its data through a well established communication link. To do so, we reduce this problem to a well-known optimization problem called Set Covering Problem (SCP) [15] that acts based upon a preprocessed set of good quality links determined by path loss propagation estimates. The model is further extended to consider multiple hop communications and smart meters coverage redundancy, in case a DAP fails. This reduction is possible because the communication load between meters and DAPs is negligible. Therefore, the main objective of a Smart Grid planning is to place DAPs so that all smart meters are covered. By solving the SCP it is possible to obtain the minimum number of DAPs needed, as well as their deployment positions. As this work evolved, a number of heuristics were proposed to solve and optimize the modeled problem.

The remaining of this work is organized as follows. Section 1.1 details Smart Grids application and explains the concept of an AMI. Related work is presented in Section 1.2. Chapter 2 describes the mathematical formulation of the Set Covering Problem and how the Smart Grid planning with multiple hops and considering minimum redundancy is reduced to it. Chapter 3 explains the theoretical basis for developing our heuristic, named MOSKOU, which is then explained in Chapter 4. Chapter 5 presents our results for multiple types of instances. Finally, Chapter 6 presents conclusions and future work.

1.1 Smart Grids

Since its creation, there have not been considerable changes on the current electrical grid's core functionality. Nonetheless, with the evolution of computer systems and networks in the XXI century, it was natural to expect that the electric grids suffered a modernization and embodied these new technologies [17]. To this intent, the Smart Grid architecture was proposed. Smart Grids refer to the use of modern communication and information techniques to increase energy distribution efficiency and reliability [22]. The National Institute of Standards and Technology (NIST) [1] proposed a conceptual model for Smart Grids communication that determines seven different modules that interconnect with each other. These modules and their relations are presented in Figure 1.1. The power generation module is composed of power plants that both generate and stores energy. The generated energy reaches the consumers by traveling through the transmission and distribution lines. In Smart Grids, both transmission and distribution lines become more active, exchanging information with consumers' smart meters, energy market and operations modules. Service providers are responsible for consumers charging and offers third party services to consumers. The energy market balances the energy offer and demand by integrating the power generation, service provider and operations modules. Finally, the operations module communicates with every other module, collecting data that allows it to efficiently manage the grid.

Each Smart Grid module possesses different actors that connect with other modules, creating interconnected networks. These networks possess their own specific functionalities, protocols and application requirements (e.g. maximum latency and minimum network bandwidth). This work focuses on the Advanced Metering Infrastructure which provides connection between consumers and the grid. The AMI is better explained in the following section.

1.1.1 Advanced Metering Infrastructure (AMI)

In a Smart Grid, the Advanced Metering Infrastructure (AMI) [28] is responsible for measuring, collecting and analyzing consumers' energy usage. Service providers are able to obtain detailed information of the power consumption of its users, ensuring a greater control in the power distribution and avoiding energy waste. The AMI allows consumers to obtain information regarding the electrical grid status, verifying power rates and consumption in almost real time.

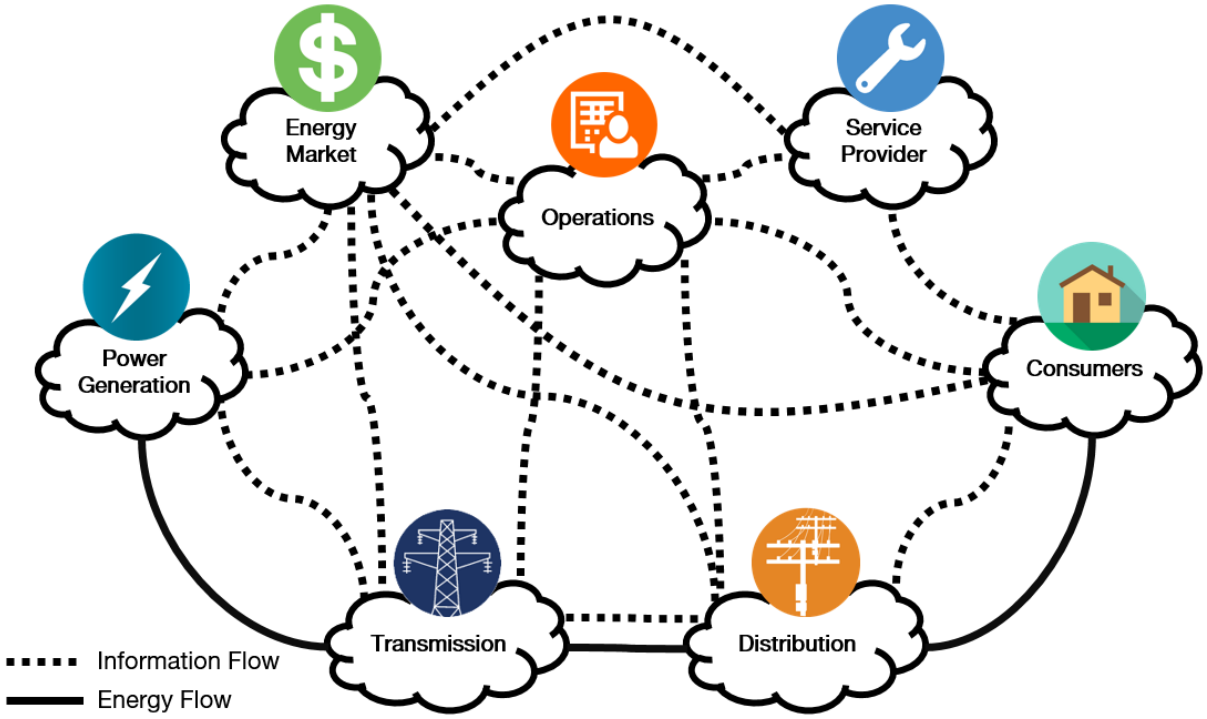


Figure 1.1: Smart Grid modules and their relations.

In order to do so, the AMI employs Data Aggregation Points (DAPs) as intermediary nodes that provide communication coverage to smart meters. A DAP can communicate with multiple smart meters in the neighborhood where it has been deployed. Consumer data received by a DAP is forwarded to the service provider's processing center where it can be analyzed. The data traffic between meters and provider is characterized by the exchange of short messages. In an AMI, smart meters generate negligible amount of load on the network, of the order of a 2400-byte packet every 240 minutes, as indicated by NIST [1]. This low amount of communication load is one of the main reasons why this work's planning model is based on a coverage optimization problem, as will be explained in Chapter 2. Additionally, through DAPs, the processing center is capable of sending commands and requisitions to meters. Analogously, the communication load from provider to smart meters is also low, of the order of 25-byte packets per event, but, for some applications, it must obey the latency limit of 1 minute.

To efficiently connect the smart devices without excessive costs for the communication infrastructure is one of the main challenges of an AMI planning. In this work, we consider that the communication between AMI devices occurs via the wireless medium in order to reduce costs [29]. As mentioned in Chapter 1, for the communication between meters and DAPs, short and medium distance communication standards such as IEEE 802.15.4 and IEEE 802.11 can be used, whereas between DAPs and provider, longer range

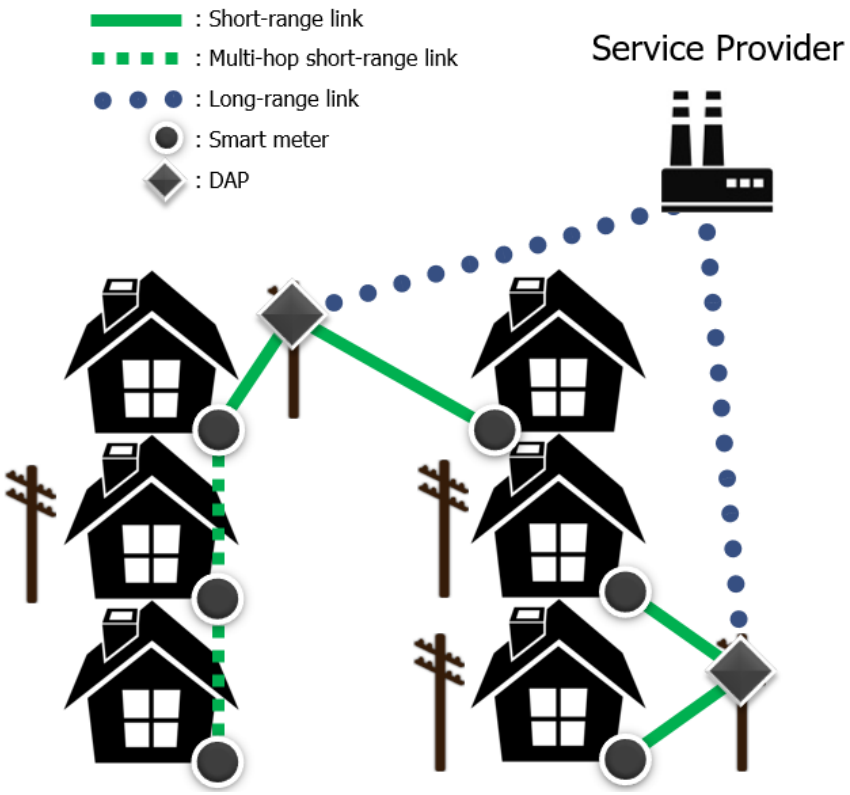


Figure 1.2: An example of advanced metering infrastructure.

technologies such as GPRS, 3G cellular network, LTE, 4G or IEEE 802.16 (WiMAX) are recommended [12]. Figure 1.2 shows an example AMI. The meters of each residence, represented by circles in this figure, connect to a DAP that forwards the meter generated data to the provider's processing center using a long distance technology. The DAPs, represented as black diamonds, can be deployed on any electric pole, as shown in the figure.

This work assumes that the communication between meters and DAPs happens exclusively through the wireless medium. Communication between DAPs and power distribution centers is out of the scope of this work. It is considered that all DAPs are capable of forwarding data generated by smart meters.

1.2 Related Work

Souza *et al.* propose a positioning mechanism for DAPs in an RF-Mesh network [23]. Their work introduces an algorithm that identifies the best position to install these DAPs based on the number of hops obtained from the Breadh-First Search, Dijkstra and Bellman-Ford methods. The best positions are obtained through an exhaustive search

that compares the number of hops for each possible installation site. Additionally, this algorithm allows the positioning of more than one DAP. In this case, the K-means clustering method is executed previously, dividing the meters into clusters. For each cluster, the exhaustive search is executed considering the closest positions to its center. The authors also propose a mathematical formulation for the problem and are able to solve it through Binary Linear Programming, having as input the number of DAPs that will be used. The disadvantage of this work is the need to previously indicate the number of DAPs that must be used.

Finding the least number of DAPs needed and their positions is not trivial. Such number is considered in the work of Aalamifar *et al* [2], which proposes a reduction of the DAP positioning problem to a facility location problem [8] and a heuristic to solve it based on the K-means algorithm. The reduced problem focuses on positioning the least number of DAPs on possible installation sites while considering both installation and transmission costs. In addition to finding the least number of required DAPs, the proposed problem also focuses on minimizing the communication's link path loss between meters and DAPs. However, the mathematical formulation of this condition makes the problem computationally complex, which is an obstacle when searching for the optimal solution, especially in city-sized instances that may contain thousands of meters and possible installation sites to consider.

The simulation of Souza *et al* considers a topology containing 67 smart meters. The biggest instance size on which the heuristic of Aalamifar *et al* was submitted contains a total of 17121 smart meters. In this work our biggest instance contains a total of 29002 smart meters and 12140 possible DAP installation sites. In order to reduce the problem's complexity, the solution proposed in this work executes a pre-processing which filters the links between meters and possible installation sites thus creating a subset of reliable links. Consequently, there is no need to formulate the path loss minimization since the coverage information is based on this pre-processed subset. This pre-processing allows the DAP positioning problem to be reduced to a simple Set Covering Problem. The pre-processing and reduction are addressed in the following chapter.

Chapter 2

The Set Covering Problem: Formulation and Reduction

This chapter details the mathematical formulation of the Set Covering Problem (SCP) and how it was modified to consider redundancy values for smart meter coverage. Next, it explains how the DAP planning in Smart Grids is reduced to the SCP.

2.1 SCP Mathematical Formulation

The classic SCP is described as follows: given a set M, of size m, and n subsets $S_j \subseteq M$, where $j \in N = \{1, ..., n\}$, each containing a non-negative cost c_j, the objective is to select one or more subsets S_j, such that each element of M belongs to at least one of these subsets, while minimizing the sum of the costs. In this work we create a new mathematical formulation that relates each element of M with a minimum value of subsets that must cover it. This value is indicated as the b_i variable. The mathematical formulation of this problem is as follows:

$$
\begin{aligned}
& min \sum_{j=1}^{n} c_j \cdot x_j \\
\text{subject to:} \quad & \sum_{j \in N} a_{ij} x_j \geq b_i, \forall i \in M \quad (1) \\
& x_j \in 0, 1, \forall j \in N. \quad (2)
\end{aligned}
$$

The x_j decision variable is equal to 1 if the S_j subset belongs to the solution and 0 if not. The a_{ij} coefficient is equal to 1 if the element i belong to S_j and 0 otherwise. The matrix $A = (a_{ij})$, $i = 1, ..., m$, $j = 1, ..., n$, is called coverage matrix. In the coverage matrix each line represents an element to be covered and each column represents a subset.

A =

	0	1	2	3
0	1	0	0	0
1	0	1	0	1
2	0	0	1	1

B =

	0	1
0	1	1
1	1	1
2	1	1

Figure 2.1: A *Set Covering Problem* example considering a minimum redundancy of 1 unit.

A =

	0	1	2	3
0	0	1	0	0
1	1	1	0	1
2	0	0	1	1
3	1	1	0	1
4	1	0	1	1

B =

	0	1
0	1	2
1	2	2
2	2	2
3	2	2
4	2	2

Figure 2.2: A *Set Covering Problem* example considering a minimum redundancy of 2 units.

Equation 1 guarantees that every element i of M is covered by at least b_i subsets. In the classic SCP, $b_i = 1, \forall i \in M$, indicating that all elements must be covered by at least 1 subset in the solution, but this value may be increased to require higher levels of coverage. Restriction 2 guarantees that x_j can be only 0 or 1. The minimization function includes the c_j variable which indicates the cost of including the j-th subset in a solution. When c_j is equal for all $j \in N$, the problem is known as Unique Cost SCP [25].

Figure 2.1 exemplifies an SCP coverage matrix considering $b_i = 1, \forall i \in M$, indicating that elements must be covered by at least 1 subset. Notice that for this example, subsets 0, 1 and 2 cover all elements of the problem, thus representing a feasible solution. The objective of the SCP, however, is to minimize the number of subsets in the solution. An optimal solution for this instance would be to select subsets 0 and 3.

Figure 2.2 shows another example of SCP but considering $b_i = 2, \forall i \in \{1...4\}$ and $b_0 = 1$. To solve this problem, lines 1, 2, 3 and 4 must be covered by 2 different columns whilst line 0 must be covered by only one. For this example, an optimal solution is choosing columns 1, 2 and 3.

2.2 Reduction: The DAP planning problem as an SCP

We can reduce the DAP planning problem to the SCP by looking at the smart meters as elements to be covered and the possible DAP installation sites as subsets containing some of those elements (i.e., containing all the meters that would be covered if a DAP was installed at that site). In the examples of Figures 2.1 and 2.2, meters would be represented by lines of the coverage matrix, while electrical poles (or other available installation sites) would be represented by columns. The B array is created based on the input redundancy value and coverage information. The values of the B array correspond to the number of installation sites (subsets) that must cover each meter (element). All positions of the B array are initialized as the input redundancy value. Nonetheless, if a smart meter cannot be covered by a quantity of DAPs equal or greater than the input redundancy, its associated value in the B array must correspond to the greatest number of DAPs that can cover it, ensuring the solution's feasibility. To better illustrate the B array creation, assume that a redundancy value of 3 is stipulated while planning an instance that contains a total of 3 meters ($M1$,$M2$ and $M3$). Due to its positions, each meter can be covered by a different set of installation sites. Assume that the first meter ($M1$) can be covered by 5 different installation sites, the second ($M2$) by 2 and the third ($M3$) by none. The corresponding value in the B array for these meters are, respectively, 3, 2 and 0. This is equivalent to saying that a feasible solutions corresponds to a configuration in which meter $M1$ is covered by 3 or more different DAPs, meter $M2$ by 2 and $M3$ by 0. If all values in the B array were set to 3 instead, there would be no feasible solution because the model would consider that meters $M2$ and $M3$ must be covered by 3 or more subsets, which is not possible. For this reason, if the input redundancy cannot be achieved by a meter, its value in the B is set to the maximum number of installation sites that is able to cover it.

In order to create the SCP it is necessary to estimate which meter is capable of establishing a reliable communication with which installation site assuming a DAP is deployed on it. This estimate is important because it dictates how close the SCP is to a real scenario. If this estimate is poorly calculated, the optimal solution of the generated SCP will be unreliable. As mentioned in Section 1.2, we execute a pre-processing to determine a subset of reliable links that will be used to create the SCP. In this way, it is not necessary to extend the SCP with a path loss minimization factor, since only reliable links which guarantee communication are considered. Therefore, the optimization problem itself remains simple. The communication link estimate is presented in the next

section.

2.3 Communication Link Estimation Calculation

The central metric to evaluate a wireless communication between a DAP and a smart meter is the Successful Delivery Rate (SDR) [16] of packets. The SDR estimation is obtained as a function of several parameters, such as the path loss, signal to noise ratio (SNR), modulation technique, chosen technology, transmission power, antenna gains and receiver sensitivity. The SDR is also associated with a propagation loss model, which must be adequately chosen to correctly represent the AMI environment.

There is a considerable number of propagation models in the literature. To be applicable for an AMI, the propagation model must befit the frequency ranges and transmission powers of both smart meters and DAPs. To this extent, different propagation models were studied such as the Okumura-Hata [19, 13], Hata COST 231 [5], the Walfisch-Ikegami [27] and, finally, the Extended Hata-SRD [6]. The first two models are applicable only at distances that exceed 1 km. The third model requires many parameters that are difficult to obtain in practice, such as the average width of streets, buildings separation distance *etc.* The most suitable propagation model studied was the Extended Hata-SRD, which is ideal for short-range devices (up to 100 m). This model is also characterized as a non line-of-sight model and takes a scenario type as input. There are 3 distinct types of scenarios: urban, suburban and rural, as recommended by ITU-R SM. 2028-1 [3]. Urban scenarios are characterized by countless sources of noises and interference plus a large number of buildings that hinders the signal propagation. Rural scenarios are the opposite, with practically no buildings and little source of noises/interference. The suburban scenario possesses characteristics of both urban and rural scenarios.

In this work, we use the Extended Hata SRD as our propagation model. With a propagation model, the SDR can be obtained from the Bit Error Rate (BER). The BER between two devices is calculated as a function of six input variables.

$$BER = f(sce, tech, power, h1, h2, d) \quad (1)$$

In equation 1, *sce* stands for the chosen scenario and determines the average noise and expected attenuation. The variable *tech* represents the parameters of the employed transmission technology, including its modulation and data rates; *power* is the transmission power in dBm of the devices; $h1$ and $h2$ represent the antenna heights of both transmitter and receptor devices; d is the distance between devices. The BER calculation

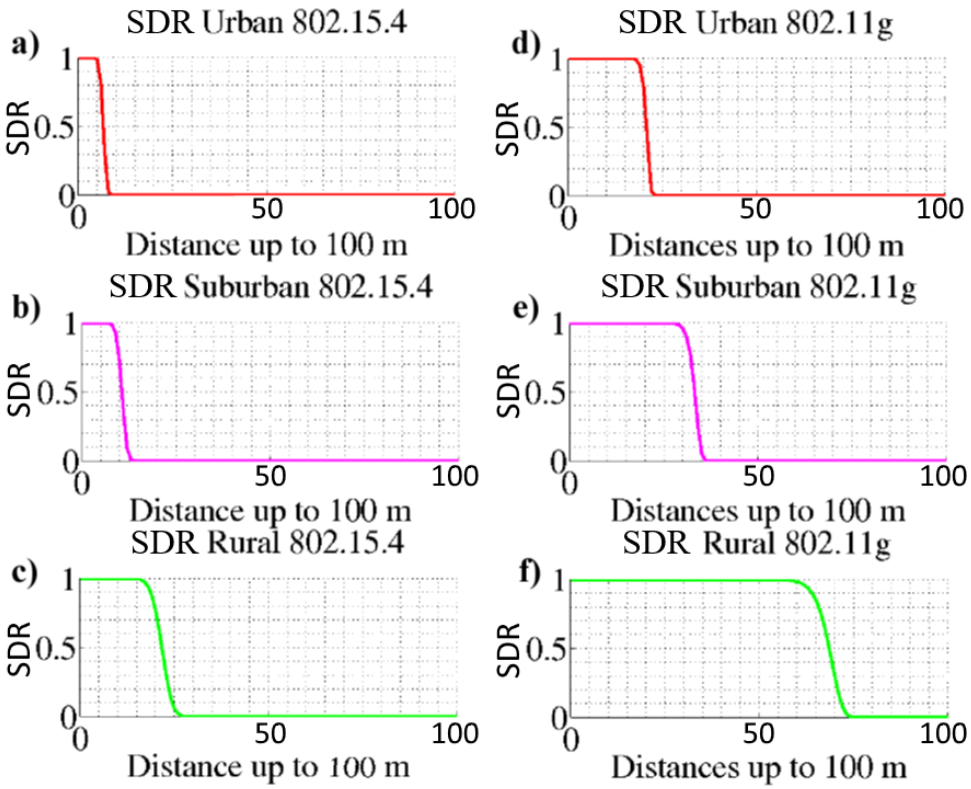


Figure 2.3: Successful delivery rate per distance for urban, suburban and rural scenarios and IEEE 802.15.4/802.11g devices.

is well-known in the telecommunications field and can be obtained in the literature such as in [24]. Further details of the BER calculation are shown in Appendix C

With the BER calculated, the next step is to calculate the Successful Delivery Rate (SDR). Equation 2 shows the SDR calculation, where n is the size of a packet in bits. The SDR represents the chance that a packet is transmitted without error. Therefore, it can be obtained as the chance that every bit is correctly transmitted. For this reason, the complementary probability of BER is raised to the power of n, indicating that all n bits must be correctly transmitted.

$$SDR = (1 - BER)^n \quad (2)$$

Figure 2.3 shows the SDR values for IEEE 802.15.4 and IEEE 802.11g devices for all types of scenarios. For IEEE 802.15.4 devices we considered a 0 dBm transmission power and data rate of 250 kbps. For IEEE 802.11g devices we considered 20 dBm transmission power and data rate of 6 Mbps. The chosen power values and data rates are typical of smart grid devices that employ these technologies. We can see that the SDR varies considerably for different types of scenarios.

Table 2.1: SDR for IEEE 802.15.4 in all scenarios.

	SDR Urban	SDR Suburban	SDR Rural
6 m	**0.99**	1.00	1.00
7 m	0.81	1.00	1.00
8 m	0.31	1.00	1.00
9 m	0.02	0.99	1.00
10 m	0.00	**0.92**	1.00
11 m	0.00	0.71	1.00
19 m	0.00	0.00	**0.95**
20 m	0.00	0.00	0.88

Table 2.2: SDR for IEEE 802.11g in all scenarios.

	SDR Urban	SDR Suburban	SDR Rural
19 m	0.99	1.00	1.00
20 m	**0.95**	1.00	1.00
21 m	0.78	1.00	1.00
31 m	0.00	0.96	1.00
32 m	0.00	**0.90**	1.00
33 m	0.00	0.77	1.00
64 m	0.00	0.00	0.93
65 m	0.00	0.00	**0.90**
66 m	0.00	0.00	0.85

2.3.1 Successful Delivery Rate Distance Limit

We consider that the subset of reliable links are composed of links with SDR superior to 90%. The value of 90% was chosen precisely to filter unreliable links. This value can be easily changed and directly impacts the SCP's solution accuracy. For example, if such limit is lowered to 70% the planning will consider that a DAP are able to cover farther meters and consequently the number of DAPs on the solution tends to drop when compared to a solution with a SDR limit of 90%. However, the increased range of communication provided by the decreased SDR limit may correspond to a poor, or even nonexistent, communication link in real scenarios that are not well represented by this model. Tables 2.1 and 2.2 show the SDR values per distance for both IEEE 802.15.4 and IEEE 802.11g devices. The maximum range of a DAP is determined by the maximum distance in which the SDR remains above 90%. In this way, the DAP range for IEEE 802.15.4 devices are 6 m, 10 m and 19 m for urban, suburban and rural scenarios respectively. The IEEE 802.11g standard, however, is able to achieve the farther distances of 20 m, 32 m and 65 m. It is important to highlight that this table was generated considering the base transmission rates of both technologies as well as transmission powers of 0 dBm and 20 dBm, for IEEE 802.15.4 and IEEE 802.11g respectively. The transmission power directly

impacts the BER calculations, and by increasing it, so does the successful delivery rate and, consequently, the DAP communication range.

2.4 Creating the SCP

As explained in the previous sections, a subset of reliable links can be obtained based on the SDR estimated for a given distance. Based on these estimates it is possible to determine a maximum radius around a DAP within which meters can establish a reliable connection to it. To reduce the DAP positioning problem to an SCP it is necessary to obtain the geographical positions of meters and installation sites in order to create the coverage matrix. For each installation site, we verify which meters are within a DAP's communication range if it is deployed at it. Meters that satisfy these conditions are represented by 1 (0 is assigned otherwise). Figure 2.4 exemplifies this process. The communication ranges are represented by dashed circles, meters are represented by filled circles and installation sites by filled diamonds. The resultant coverage matrix is also illustrated. Notice that the meters inside the reliable transmission range of an installation site are assigned the value 1 in the correspondent position of the coverage matrix, while 0 is assigned otherwise.

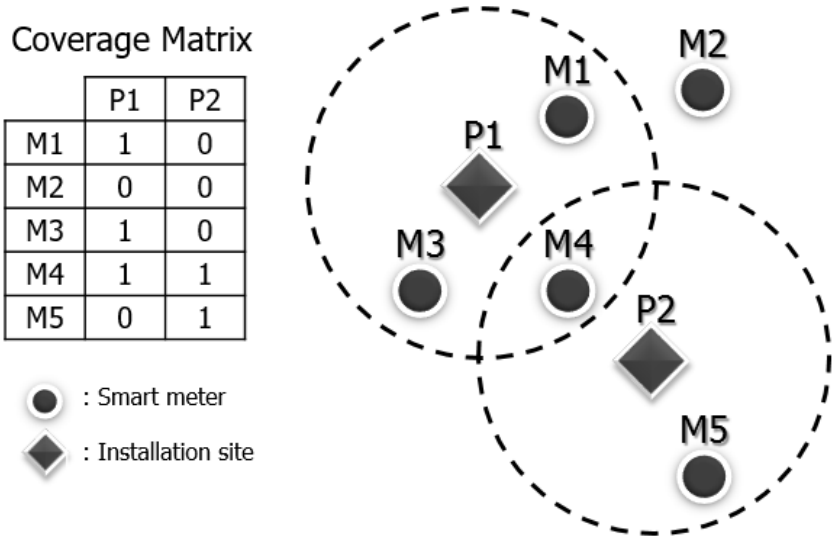


Figure 2.4: An example of a coverage matrix based on the meters and possible installation sites.

Analogously, it is possible to construct the coverage matrix considering the possibility of multihop communication. In this case, we have to consider that meters are capable of acting as relay nodes, allowing DAPs to reach meters which, otherwise, would be unreachable. Given an installation site and its directly reachable meters, we consider

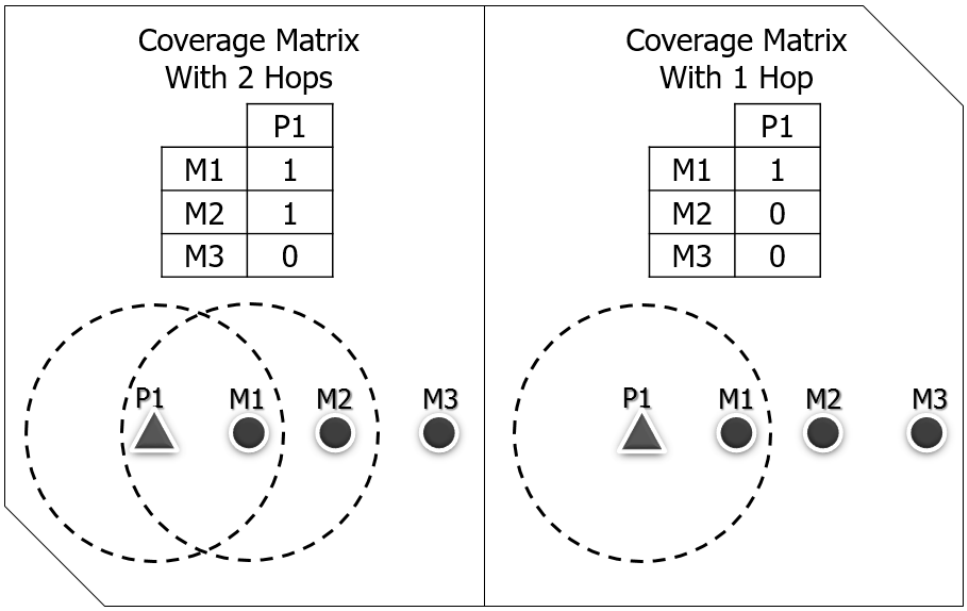


Figure 2.5: Coverage matrix comparison with and without multiple hops.

that this site also reaches all meters that have these reachable meters as neighbors. This process can be repeated for an arbitrary number k of times, potentially adding more meters to the coverage set of each installation site and requiring up to k+1 hops for the meters to reach their DAPs. Figure 2.5 compares the coverage matrices built considering only 1 hop communication and up to 2 hops in the same scenario. Using 2 hops, meter M1 can serve as a relay for installation site P1 to also cover M2.

Chapter 3

Theoretical Basis for the Heuristic's Construction

The SCP can be solved by either using a linear programming solver, such as the CPLEX [14] and GLPK [18], or employing a heuristic. Solvers return the optimal solution but may need a great amount of memory and processing capability depending on the size and complexity of the problem. On the other hand, heuristics are often capable of obtaining good solutions while requiring less computational resources and in shorter execution time.

Nevertheless, results of this work have shown that SCP instances based on the smart grid planning problem possess distinct characteristics that allows them to be more easily solved due to its typical small density, as will be explained later in this chapter. However, the amount of required memory is an obstacle to finding optimal solutions for big instances. This chapter analyzes the behavior of SCP instances regarding the required amount of RAM memory and execution time based on their dimension (i.e. size) and it's density (i.e. percentage of 1's on the coverage matrix). This analysis serves as basis for our heuristic's construction, which adopts a divide-and-conquer approach, splitting the problem in smaller ones that are optimally solved. Our proposed heuristic is named MOSKOU and, for simplicity, is addressed as such from now on.

To optimally solve each sub-problem generated by the MOSKOU heuristic, we employed the GLPK solver. For this reason, all results in this chapter were obtained using the GLPK and both memory and execution evaluation are linked to this specific solver. If another solver is to be used, new results must be gathered to adapt the MOSKOU heuristic to it. All the following tests were executed on a Windows 64-bits, 3 GHz Intel Core i5 CPU with 8 GB of RAM memory. In order to guarantee the feasibility of this

evaluation, we imposed a time limit of 500 seconds for each execution of the solver.

3.1 SCP Dimension: Memory Impact

Prior to starting to solve the Set Covering Problem, the solver needs to allocate an initial amount of memory sufficient to attend its mathematical restrictions. For the SCP, these mathematical restrictions are linked to the number of elements to be covered and the subsets that cover them. As mentioned in Chapter 2, the elements to be covered are the smart meters whereas the subsets are the possible installation sites' coverage range. Therefore, an initial amount of memory can be predicted based on the number of meters and installation sites alone. It is important to notice that this amount may increase according to the coverage matrix's density, which will be analyzed later.

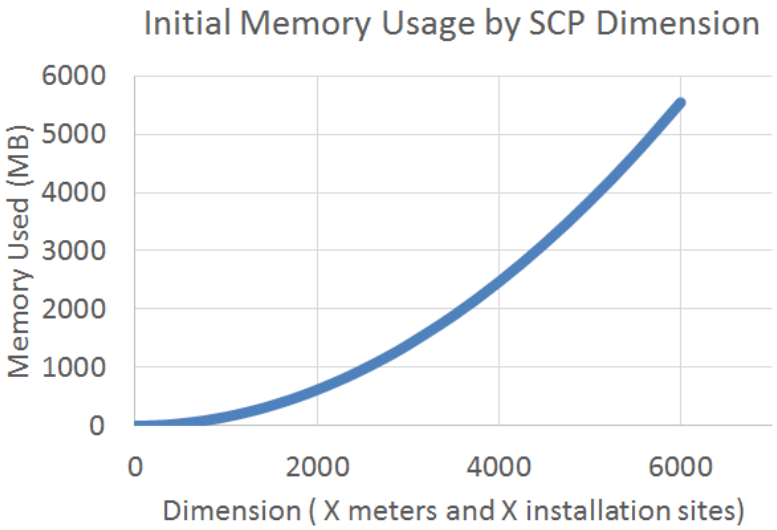


Figure 3.1: Initial memory consumption per dimension. The X-Axis corresponds to the number of meters and installation sites.

Figure 3.1 shows the increase of the initial memory consumption according to the SCP's dimension. The plotted values were obtained experimentally. The Y-Axis indicates the initial amount of memory in megabytes for a problem containing a number of meters and installation sites equivalent to the value on the X-Axis. For example, a problem containing 3000 meters and 3000 installation sites requires an initial amount of memory of approximately 1400 MB.

Figure 3.2 compares the amount of memory required as a function of only the number of meters (X-Axis), but parametrized by the number of installation sites (the figure shows curves for 1000, 2000, 3000 and 4000 installation sites). With a fixed number of installation

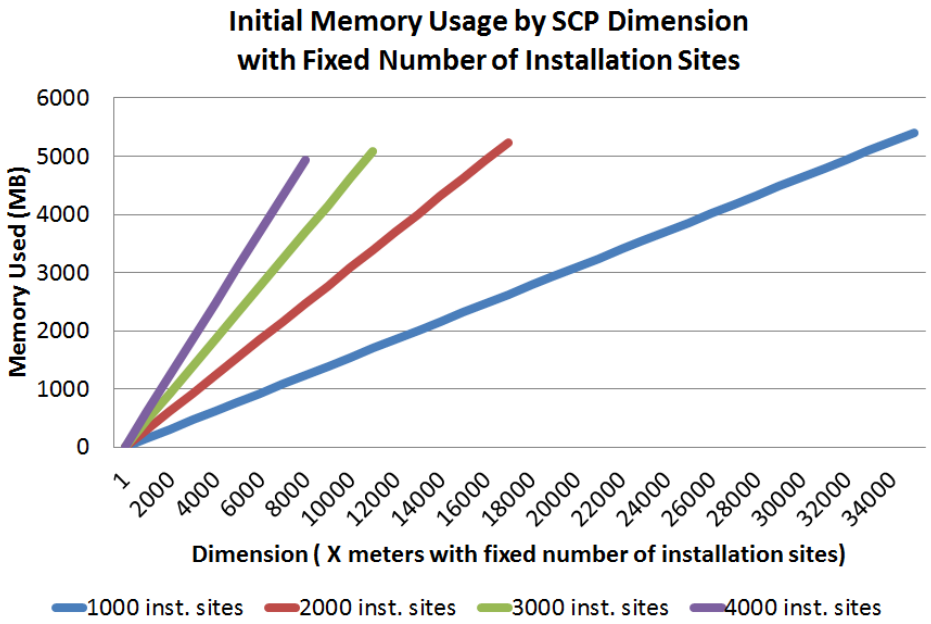


Figure 3.2: Memory consumption per dimension with fixed number of installation sites.

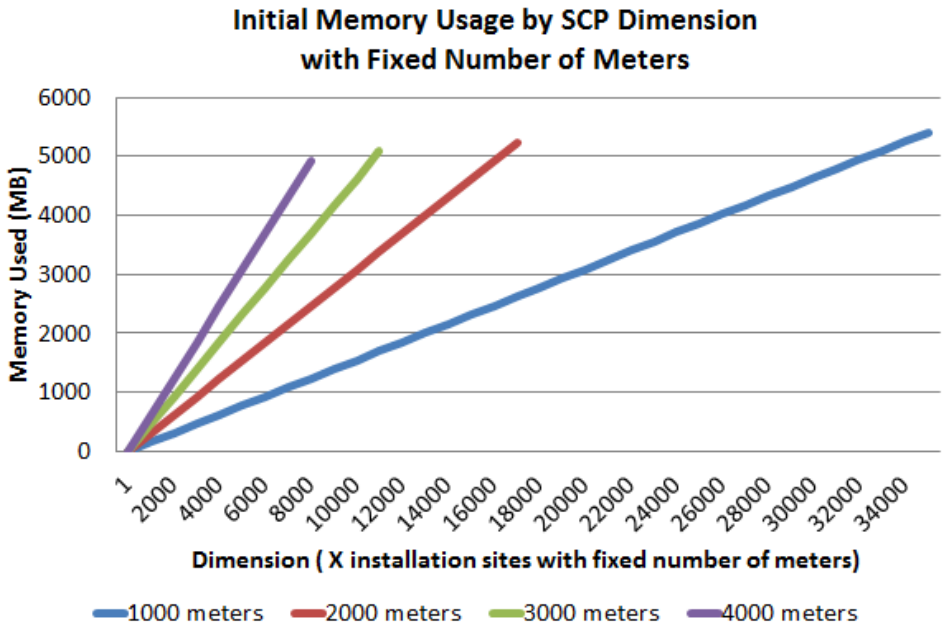


Figure 3.3: Memory consumption per dimension with fixed number of meters.

sites, the function grows linearly with the number of meters. By fixing the number of meters instead, the behavior is nearly the same with negligible variation as can be seen in Figure 3.3.

Based on these data, we determined an approximated function to estimate the initial amount of required memory for a given problem based solely on the problem's dimension. This function is useful, for example, to determine if a given instance is solvable according

to the computer's memory limit and is employed on the MOSKOU heuristic's division phase. Algorithm 1 shows how the estimation function is computed. The values of the coefficients a, b and c were obtained by fitting a trending line to the experimental data shown in 3.1 and are shown in Table 3.1 for 32 and 64 bit systems (64 bit system use more memory because the sizes of certain data types are larger). To evaluate our estimate's accuracy, we generated 100 random instances with the number of installation sites and meters varying from 500 to 5000 and compared the gap between the real used memory and its estimate. We obtained an average gap of 0.2% for these instances with a maximum gap of 1.2% and a minimum of 0.03% which are negligible amounts for the purpose of evaluating the feasibility of solving a given instance in a certain system.

input : Num. of meters, Num. of installation sites
output: Memory estimation
1 $minVal \leftarrow$ min(Num. of meters, Num. of installation sites) ;
2 $maxVal \leftarrow$ max(Num. of meters, Num. of installation sites) ;
3 memEst $\leftarrow a \times minVal^2 + b \times minVal + c$;
4 Memory estimation $\leftarrow memEst \times (maxVal/minVal)$;

Algorithm 1: Initial memory estimation function.

Table 3.1: Memory estimation coefficients for 32 and 64 bit systems.

	32-bits	64-bits
a	10.727×10^{-5}	15.368×10^{-5}
b	10.449×10^{-4}	16.504×10^{-4}
c	92.379×10^{-3}	-91.447×10^{-3}

3.2 SCP Density: Impact on Memory and Solving Time

The density of an SCP is defined as the percentage of 1's in the coverage matrix. For example, the coverage matrix of an SCP with 20 elements (meters) and 30 subsets (installation sites) has 600 (i.e. 20 x 30) pairs. If that instance has a density of 20%, that means 120 of the 600 pairs are set to 1. As will be demonstrated, the SCP's density is directly associated to its complexity, which impacts on its memory consumption and solving time. To evaluate this impact, we generated random SCPs instances with varying densities from 1% to 100% in an additive fashion. In other words, for an instance with 100 meters and 100 installation sites, an initial SCP is randomly generated with 1% density, i.e. 100 random pairs of its coverage matrix are set to 1. Next, the instance density is incremented by 1% by randomly replacing zeros with ones in the coverage matrix. The process is repeated until 100% density is obtained. Notice, however, that the density

value only determines the number of 1's in the coverage matrix but does not contain information on its distribution. For example, Figure 3.4 shows two coverage matrices with the same amount of elements (5), subsets (5) and density (20%). Notice that in the coverage matrix of Figure 3.4(a) all 1's are distributed on only one subset, denoting that column 0 can cover all elements while the remaining columns are not able to cover any. However, in Figure 3.4(b) the 1 values suffer a more balanced distribution in which all subsets are capable of covering one distinct element. Since it's impracticable to analyze all density distributions, we generated our instances according to Algorithm 4 depicted in Appendix A.

	0	1	2	3	4
0	1	0	0	0	0
1	1	0	0	0	0
2	1	0	0	0	0
3	1	0	0	0	0
4	1	0	0	0	0

(a) Figure A

	0	1	2	3	4
0	1	0	0	0	0
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	0
4	0	0	0	0	1

(b) Figure B

Figure 3.4: Two coverage matrices with the same density but different distributions.

Figure 3.5 shows the solving time for a SCP with 100 meters and 100 installation sites. Since the instances were randomly created, we repeated the tests 5 times with different seeds. The middle curve is the average result of all instances, whereas the top and bottom curves are the maximum and minimum values. While the curves present some statistical fluctuation, it is possible to notice that the solving time is short for densities up to 5% and rises until approximately 22% to decrease right afterwards. While, at first, increasing the density makes the problem harder, at some point the installation sites are capable of covering a greater portion of meters, thus simplifying the solver's choices. One of the simplest case scenarios happens with 100% density, i.e, when every installation site covers every meter. Therefore, the optimal solution comes down to choosing only one installation site, which is simple and virtually instantaneous.

Figure 3.6 shows the same kind of experiments, but considering instances with 150 meters and 150 installation sites. By increasing the number of meters and installation sites by 50% each, the average solving time peak reached the stipulated 500 seconds limit. However, for densities lesser than 5% and higher than 55% the optimal solution was found in less than 1 second.

Figure 3.7 compares the amount of memory used on both the 100x100 and 150x150 in-

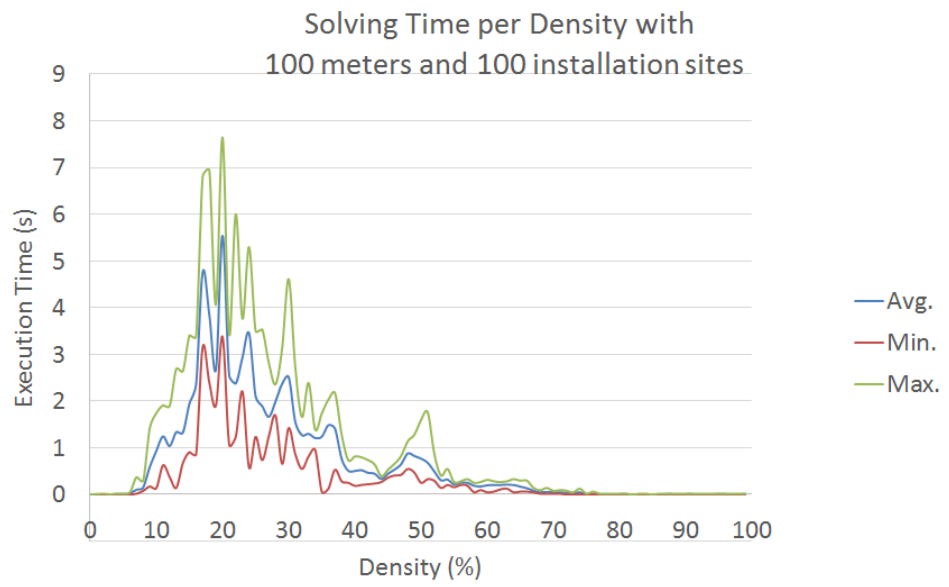


Figure 3.5: Solving time per density on a 100x100 SCP. A time limit of 500 seconds was set.

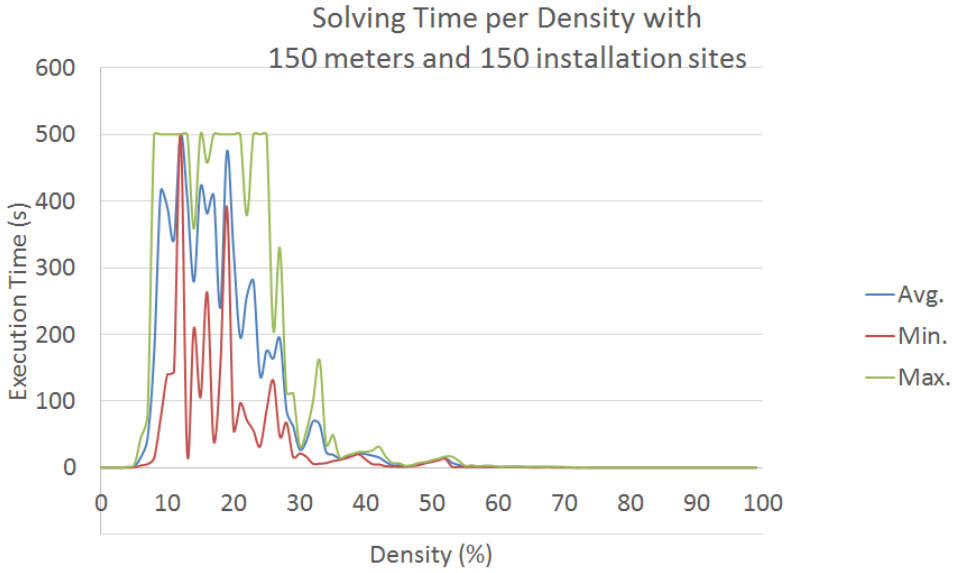


Figure 3.6: Solving time per density on a 150x150 SCP. A time limit of 500 seconds was set.

stances mentioned above. It is important to notice that even though the solving time drops as the density gets closer to 100%, the memory consumption rises linearly. Additionally, with 100 meters and installation sites, the maximum solving time reached approximately 8 seconds whereas with 150 it exceeded the stipulated 500 seconds time limit. However, the maximum memory difference on both instances was only 3 MB. In this work we do not estimate the amount of memory based on the instance's density because typical smart grid SCP densities are small enough to barely interfere on the total required memory.

To conclude this chapter, it is important to elucidate that most SCP instances created based on Smart Grids scenarios possess tiny density values, typically lesser than 1%. For example, the largest density value, considering a maximum of 4 hops, found for the instances presented in Section 5 is only 1.07%, which was obtained for the Niterói instance. On the other hand, these instances are characterized by the high amount of meters and possible installation sites that must be analyzed. Usually, these instances can be optimally solved in small amount of time but require a great amount of RAM memory, which is the main hindrance to solve them. For this reason, the MOSKOU heuristic, explained on the following chapter, employs a divide-and-conquer approach based on the memory usage estimation.

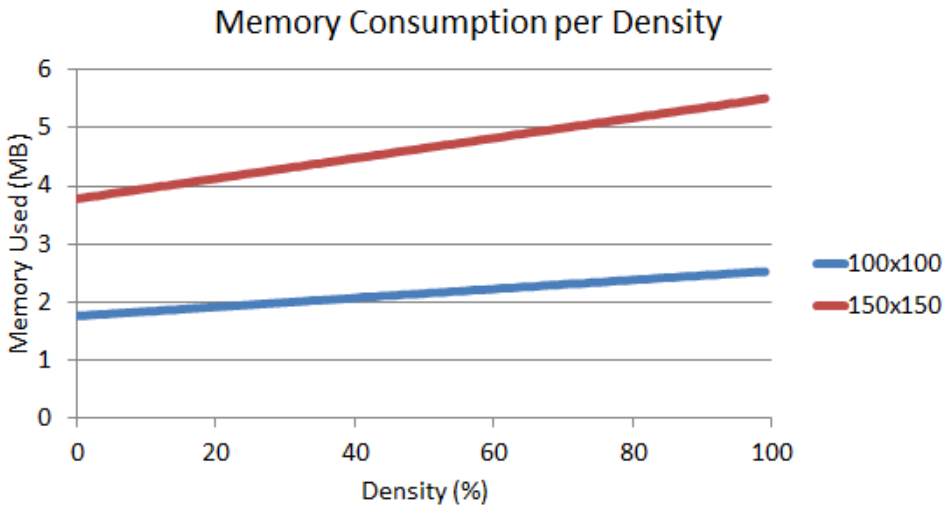


Figure 3.7: Memory consumption per density for the 100x100 and 150x150 SCPs.

Chapter 4

The MOSKOU Heuristic

The SCP instances resulted from the reduction of the DAP planning problem follow a geographical pattern, i.e. meters reachable by an installation site are limited to a region around it. Due to this characteristic, those instances belong to a particular type o SCP known as Euclidean SCP. Yelbay et al. [30] explain that Euclidean SCPs present small complexity and that the progress of solvers over the years allowed them to be rapidly solved. Still, the typical scale of the DAP planning problem requires the solvers to use very significant amounts of memory, which often prevents the optimal solution to be found. By noticing that memory consumption is often the bottleneck for solving the DAP planning problem, the MOSKOU heuristic employs the divide-and-conquer approach, splitting large, unfeasible instances into smaller sub-instances that can be solved within the available amount of memory. The individual results are then merged by a post-optimization method, composing the solution of the original instance. MOSKOU stands for `Memory Oriented Split using K-Means with post-Optimization Unification`.

Algorithm 2 shows how the MOSKOU heuristic works. The heuristic takes an instance to solve as input and a numeric value that determines the maximum amount of RAM memory to use. These values are represented as `Instance` and `MemLim`, respectively. The `Instance` input is added to the `SubInstances` set and while the `SubInstances` set is not empty, an element is chosen from it (line 4) and it is verified if the estimated amount of memory required to solve it exceeds the memory limit (line 5). The memory estimation function corresponds to Algorithm 1 of Chapter 3. If the memory estimation of an instance exceeds the memory limit, it is divided into two smaller instances by the clustering algorithm K-Means (line 6), with $K = 2$. More specifically, the K-Means algorithm acts upon the set of meters, dividing it into two subsets. For each of these subsets a new sub-instance is generated composed by the associated subset of meters and

```
   input  : Instance, MemLim
   output: Solution
 1 SolvableInstances ← {};
 2 SubInstances ← Instance ;
 3 while SubInstances ≠ ∅ do
 4 |   Inst ← GetElement(SubInstances);
 5 |   if MemoryEstimation(Inst) ≥ MemLim then
 6 |   |   SubInst1, Subinst2 ← Kmeans2(Inst);
 7 |   |   SubInstances ← SubInstances ∪ SubInst1;
 8 |   |   SubInstances ← SubInstances ∪ SubInst2;
 9 |   end
10 |   else
11 |   |   SolvableInstances ← SolvableInstances ∪ Inst;
12 |   end
13 |   SubInstances ← SubInstances − Inst;
14 end
15 Solution ← {};
16 foreach Inst ∈ SolvableInstances do
17 |   Solution ← Solution ∪ Solve(Inst);
18 end
19 Post-Optimization(Solution);
20 return Solution;
```

Algorithm 2: MOSKOU heuristic algorithm.

a all installation sites that are able to cover any of these meters.

The K-Means algorithm ensures that the split sub-instances are formed by meters and installation sites that are geographically near one another. Therefore, it is possible that the split sub-instances possess no relation to one another (e.g.: different neighborhoods of a city) which increases the probability that the joint solution is, indeed, close to the optimal. However, even if the split sub-instances are related to one another (e.g., a neighborhood split in half), a post-optimization method is applied to soften the inaccuracy imposed by the division.

The new sub-instances created after the split are added to the `SubInstances` set (lines 7,8) to be further analyzed regarding their memory consumption. Whenever the memory estimation is lesser than the limit, the analyzed instance is added to the `SolvableInstances` set, indicating that it can be optimally solved. When the `SubInstances` set is empty, each element in `SolvableInstances` is applied to the solver and their solutions joined (line 16,17). The joined solution is applied to a post-optimization method and returned as the final result (lines 19,20).

Naturally, the optimal solution of each sub-instance is not necessarily part of the

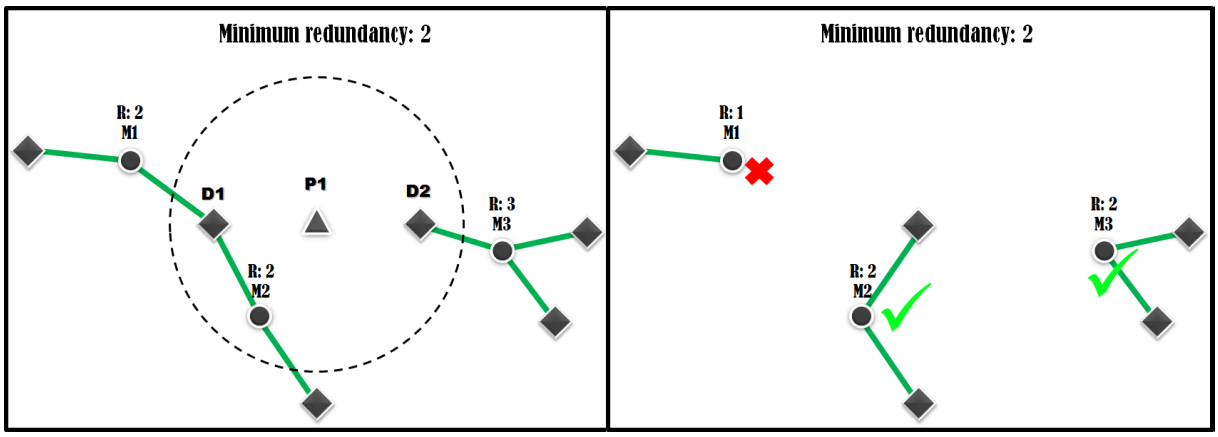
```
   input  : Solution, Redundancy
   output: Optimized Solution
 1 succeded ← true ;
 2 while succeeded do
 3 |   succeded ← false ;
 4 |   foreach installation site ∉ Solution do
 5 |   |   replaceableInstallationSites ← {};
 6 |   |   foreach selectedInstallationSite ∈ Solution do
 7 |   |   |   if CanBeRemoved(selectedInstallationSite,Redundancy) then
 8 |   |   |   |   replaceableInstallationSites ← replaceableInstallationSites ∪
   |   |   |   |   {selectedInstallationSite};
 9 |   |   |   end
10 |   |   end
11 |   |   if |replaceableInstallationSites| ≥ 2 then
12 |   |   |   foreach toReplace ∈ replaceableInstallationSites do
13 |   |   |   |   Solution ← Solution − {toReplace};
14 |   |   |   end
15 |   |   |   Solution ← Solution ∪ {installation site};
16 |   |   |   succeeded ← true;
17 |   |   |   break;
18 |   |   end
19 |   end
20 end
```

Algorithm 3: Post-optimization algorithm.

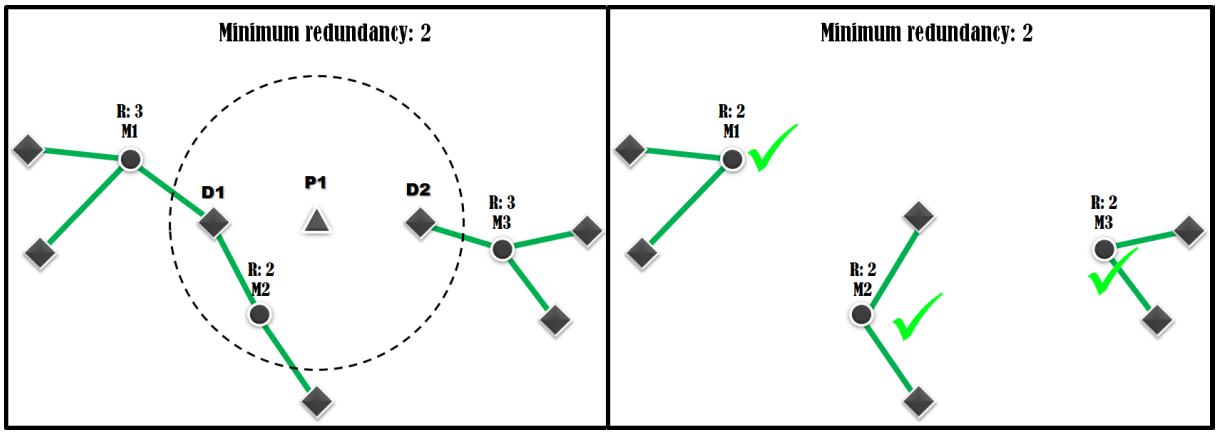
optimal solution of the complete problem. There is a high chance of unnecessary DAPs on the solution, mainly at the edges of each sub-instance. Hence, we propose a post-optimization method that aims at reducing these redundancies by finding an installation site position that can replace 2 or more already placed DAPs. Algorithm 3 shows the post-optimization pseudo-code.

The input solution comprises all installation sites selected by the first stage of the heuristic. The post-optimization method checks for each installation site that does not take part in the solution (line 4) if there is a set of 2 or more selected installation sites that can be removed if the analyzed installation site becomes selected. If such set exists (line 11), the replacement occurs (lines 12 to 15) and the process restarts from the beginning. If the number of selected installation sites is no longer reducible, the post-optimization finishes.

To determine if an installation site is removable, it is imperative that after the substitution, the redundancy of all meters that were covered by the analyzed DAPs remains equal or greater than the established redundancy. If the redundancy of one of these meters were inferior to the established redundancy, it must not further decrease. Figure 4.1



(a) An unfeasible substitution example.



(b) A feasible substitution example.

Figure 4.1: Two examples of both unfeasible and feasible DAPs substitution.

shows a visual exemplification of the above mentioned condition considering a minimum redundancy of 2. The number above each meter represents the number of DAPs that cover it (i.e. its redundancy). Links between meters and DAPs are represented by a green line. In this example, the post optimization method verifies if installation site P1 can replace 2 or more DAPs. Notice that, for the first example in Figure 4.1(a), by removing both D1 and D2 and assuming that P1 is now part of the solution, meter M1's redundancy drops below the established redundancy. Therefore the substitution is not possible. It is also important to notice that redundancy of meter M2 remains the same because installation site P1 is now in charge of covering it. At the same time, meter M3's redundancy drops to 2 which is not a problem, since the established redundancy is also 2. The example in Figure 4.1(b) shows a viable substitution in which both DAPs D1 and D2 are replaced by a new DAP placed in installation site P1. Therefore, when installation site P1 is being analyzed for this instance, the method *CanBeRemoved* returns *true* when analyzing both DAPs D1 and D2.

Chapter 5

Results

The goal of this chapter is to compare the performance of the MOSKOU heuristic with the linear programming exact method and other heuristics. The comparison takes into account the execution time, quality of the solution (i.e. the number of required DAPs) and the memory requirements. The utilized solver was GLPK [18]. All methods were executed on only one thread on a 3 GHz Intel Core i5 CPU with 8 GB of RAM memory. Instance characteristics and results are shown in the following subsections. All tests considered IEEE 802.11g devices transmitting at 20 dBm with a fixed 6 Mb/s rate. We set a solving time limit of 60 seconds for the GLPK solver. Exceptionally for the Grid instances the solver had difficulty finding the optimal solution and exceeded the time limit. When the time limit is exceeded, the solver returns the best solution found so far. In subsection 5.4.1 we discuss that the values returned from the solver when the time limit is reached have a small gap compared to the optimal solution.

It is important to highlight that a multi-threaded approach is also possible when executing the MOSKOU heuristic. However, the main hindrance when solving a Smart Grid instance is the amount of RAM memory needed. Therefore, a multi-threaded approach is not beneficial since the available memory is limited. Nonetheless, a heuristic that utilizes a distributed solver on multiple machines to increase the available amount of RAM can be implemented and is one of our recommendations for future work.

5.1 Instances

We evaluate the performance of the MOSKOU heuristic in four different instances that are explained below.

5.1.1 Florianópolis Instance

The Florianópolis instance was created based on real geographical positions of electrical poles in the city of Florianópolis, Brazil. This instance represents a power feeder on the neighborhood of Agronômica which is one of the regions addressed in the SIRIS (SImulador de Redes InteligenteS) project. Pole positions and the number of smart meters in this neighborhood were obtained from the power distribution company CELESC. Based on these values, the instance of Florianópolis contains a total of 12140 electrical poles (i.e. installation sites) and 29002 meters. Figure 5.1 shows this region. We consider this instance as an urban scenario.

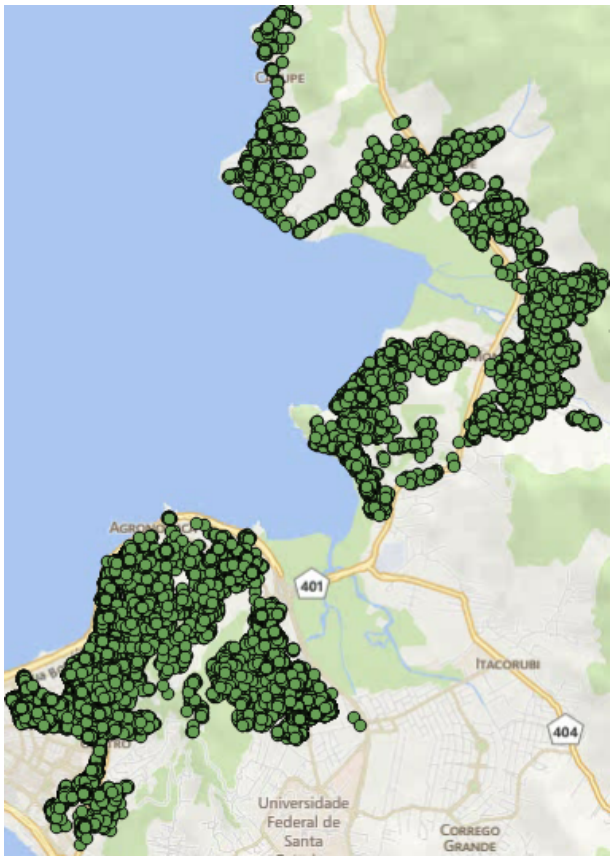


Figure 5.1: Meter distribution in the Florianopolis instance.

5.1.2 Niterói Instance

The Niterói instance is based on the real positions of streets and houses at the neighborhoods of Icaraí and São Francisco in the city of Niterói, Brazil. Meters were distributed along the streets spaced from one another by a distance varying randomly from 15 to 25 m. The distance between the electrical poles varied randomly from 30 to 50 meters.

Figure 5.2 shows this region, which contains a total of 3666 smart meters and 1030 possible installation sites. Due to its density and distribution we consider this instance as a suburban scenario.

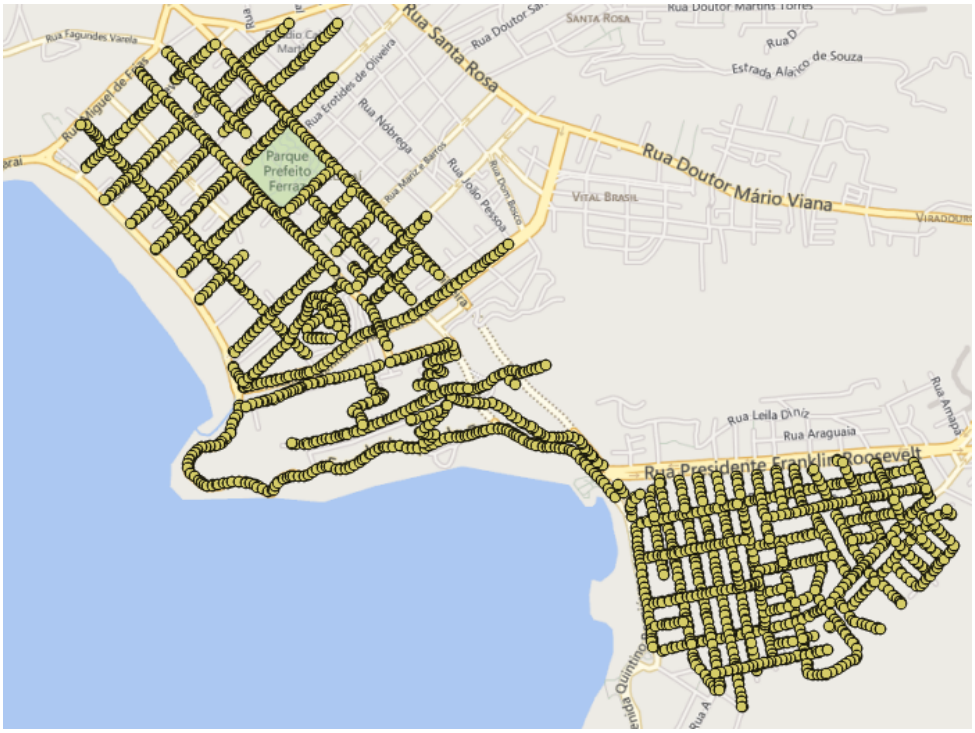


Figure 5.2: Meter distribution in the Niterói instance.

5.1.3 Urban Grid Instance

The Urban Grid instance was generated based on the average of 2000 smart meters per km^2 as indicated by NIST[1]. We consider blocks of 100x100 m^2 each containing 20 smart meters randomly placed to maintain the same proportion. Each block is separated from one another by 10 m. Each block contains a total of 36 installation sites positioned according to a grid formation. In total, the Urban Grid instance contains 8000 meters and 12200 poles (installation sites). Figure 5.3 shows an example of block in the Urban Grid instance. The Urban Grid instance, as well as the Suburban Grid instance detailed in the following subsection, have a particularity regarding the proportion of installation sites per meter. Unlike the instances of Florianópolis and Niterói, the number of installation sites on both Urban and Suburban Grid instances is greater than the number of meters. The degree of choices a solver must analyze is directly associated to the number of installation sites and has a major impact on both execution time and memory required.

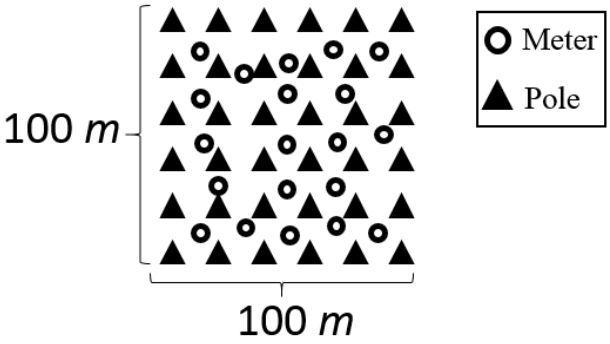


Figure 5.3: An example of block in the Urban Grid instance.

5.1.4 Suburban Grid Instance

The Suburban Grid instance was generated analogously to the Urban Grid but with an average of 800 smart meters per km^2 as indicated for suburban scenarios. Just as in the Urban Grid instance we consider blocks of 100x100 m^2, but each block now contains 8 meters randomly placed. The installation site distribution remains the same. However, to simulate a suburban scenario each block now contains 16 poles (installation sites) instead of 36. In total, the Suburban Grid instance contains 3200 meters and 4920 installation sites. Figure 5.4 shows a block in the Suburban Grid instance.

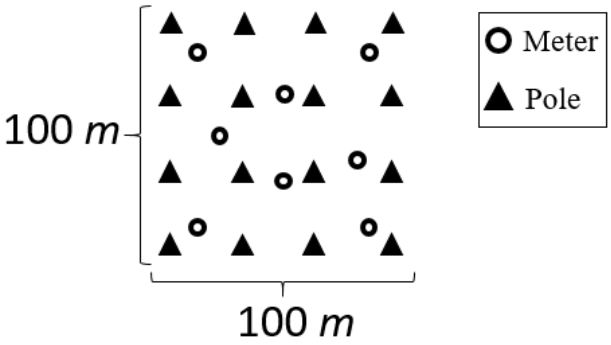


Figure 5.4: An example of block in the Suburban Grid instance.

5.2 Heuristic Comparison Results

In this section we compare the MOSKOU heuristic with two other heuristics regarding solution quality (i.e. Number of DAPs) and execution time. The first heuristic adopts a greedy approach, choosing first the installation sites that can cover the greatest amount of smart meters until all possible meters are covered. Due to it's simplicity, greedy heuristics are able to find a result in small amount of time but usually return bad solutions. The second heuristic, named Grid heuristic, was developed on the initial stages of this work

and adopts a divide-and-conquer approach that splits the problem in fixed sized cells that are optimally solved independently and its results joined afterwards. The Grid heuristic is explained in Appendix B.

Figure 5.5 compares all three heuristics for the Niterói instance. Each external column represents the maximum number of hops, which was considered to a maximum of 4 hops. Each line exhibits the solution's quality (i.e. number of DAPs) and execution time in seconds given a redundancy input, which varied from 1 to 3. All heuristics were submitted to the post-optimization method detailed in Algorithm 3. The optimal solution is displayed in bold. Green cells indicate the winner, yellow cells the second best value and red cells the worst values.

Maximum Number of Hops

Target Redundancy

	1			2			3			4		
	Greedy	Grid	MOSKOU	Greedy	Grid	MOSKOU	Greedy	Grid	MOSKOU	Greedy	Grid	MOSKOU
1	543	**528**	**528**	287	**276**	**276**	194	**178**	**178**	137	**131**	**131**
	0.053	13.04	13.25	0.224	13.40	13.65	0.266	13.84	14.13	0.335	14.43	14.84
	Greedy	Grid	MOSKOU	Greedy	Grid	MOSKOU	Greedy	Grid	MOSKOU	Greedy	Grid	MOSKOU
2	936	**923**	**923**	577	**562**	**562**	381	**368**	**368**	275	**265**	**265**
	0.04	12.87	12.91	0.218	13.34	13.54	0.277	14.27	14.47	0.351	14.27	14.62
	Greedy	Grid	MOSKOU	Greedy	Grid	MOSKOU	Greedy	Grid	MOSKOU	Greedy	Grid	MOSKOU
3	1019	**1015**	**1015**	819	**803**	**803**	569	**558**	**558**	413	**402**	**402**
	0.047	12.96	12.93	0.219	13.20	13.38	0.280	13.99	14.29	0.350	14.26	14.61

Figure 5.5: Preliminary heuristics comparison.

We can see that the greedy heuristic obtained the best execution times but, on the other hand, had the worst solutions for all results. Both Grid and MOSKOU heuristics obtained the optimal solution and in practically the same amount of time. This can be explained by the methodology under which these heuristics were created. Both Grid and MOSKOU differ only on the split method employed whenever the available amount of RAM is to be exceeded. However, the Niterói instance is relatively small and was capable of being optimally solved with the computer's available memory. Therefore, whenever an instance can be optimally solved, both Grid and MOSKOU operate in the same way, creating the problem and applying to the solver without the need to split the problem.

To efficiently compare the Grid and MOSKOU heuristics, the same experiment was run for the Florianópolis instance with the only difference being that the Greedy heuristic is omitted due to its poor solution's quality on the previous experiment. Figure 5.6 shows these results. The optimal solution for the Florianópolis instance is unknown due to

the huge amount of memory needed. We estimate that more than 50 GB of RAM are required to optimally solve it. We can verify that, for this instance, the performance of the MOSKOU heuristic surpasses the Grid's in both execution time and solution's quality. The MOSKOU heuristic was able to obtain slightly better solution qualities requiring 24.8% less execution times on average. The execution time difference gain can be explained by the number of sub-instances generated by each heuristic. The Grid Heuristic divides the problem in cells with fixed size at the beginning of it's execution. By using cells, the Grid Heuristic does not efficiently group elements. Therefore, the problem is typically split in a much greater number of sub-instances when compared to the MOSKOU heuristic. This requires a greater number of problem setups (i.e. creating the problem) and more calls to the solver (one for each sub-instance), which takes time. For the Florianópolis instance, the Grid heuristic divided the problem in 27 sub-instances, whereas the proposed MOSKOU heuristic in only 4. The efficient split generated by the MOSKOU is the great advantage of this heuristic in comparison to the Grid one. By better dividing an instance, it is possible to obtain better quality and in lesser time. The next section analyzes how the MOSKOU heuristic behaves by varying the maximum number of hops and meter redundancy.

Maximum Number of Hops

Target Redundancy

		1		2		3		4	
		Grid	MOSKOU	Grid	MOSKOU	Grid	MOSKOU	Grid	MOSKOU
1		2979	2973	1999	1992	1690	1682	1569	1559
		520.9	401.4	556.7	408.3	604.5	447.0	660.9	510.8
		Grid	MOSKOU	Grid	MOSKOU	Grid	MOSKOU	Grid	MOSKOU
2		5121	5115	3497	3493	2878	2871	2615	2609
		517.9	404.5	561.5	400.3	610.1	450.6	666.8	542.3
		Grid	MOSKOU	Grid	MOSKOU	Grid	MOSKOU	Grid	MOSKOU
3		6393	6387	4685	4681	3829	3824	3431	3427
		524.3	368.0	573.2	399.8	616.7	456.3	673.6	556.9

Figure 5.6: Grid and MOSKOU heuristics comparison.

5.3 Maximum Number of Hops and Redundancy Impact on the Solution

This section analyzes how the execution time and solution quality are affected by varying the number of hops and redundancy. For an initial experiment we consider the results of our proposed MOSKOU heuristic for the Florianópolis instance. Figure 5.7 shows these values plus a new value that corresponds to the solving time. The solving time represents the time spent by the solver to obtain the optimal solutions when the necessary memory has already been allocated. The solving time exhibited corresponds to the sum of the solving time for all sub-instances generated. For this instance, 4 sub-instances have been generated by our heuristic.

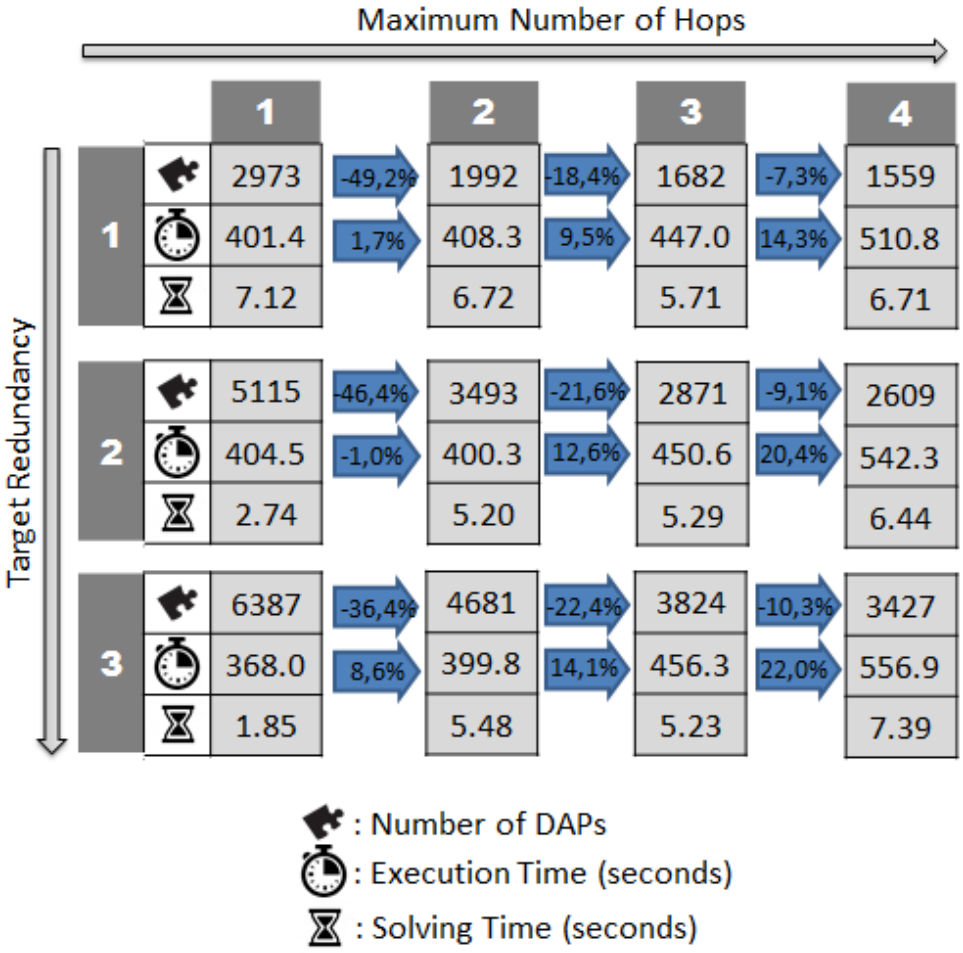


Figure 5.7: Maximum number of hops and redundancy variation analysis for MOSKOU's solutions in the Florianópolis instance.

Naturally, we can verify that the greater the number of hops, the lesser the quantity of DAPs needed to solve the problem. Nonetheless, as the maximum number of hops increases, the number of needed DAPs diminishes slower. Results also show that the

execution time rose with the maximum number of hops. This happened because to create the coverage matrix with greater number of hops, the heuristic takes more time to compute the problem because it needs to verify additional set of coverable meters that become available due to the hop increment. We can conclude that the coverage matrix creation is the main time consuming process because the solving time was small (less than 7.5 seconds for all cases) and varied little with both redundancy and number of hops. Also, the redundancy variation had inconclusive impact on the execution time. For 1 hop, by varying the redundancy from 2 to 3, the execution time was reduced in 36.5 seconds whereas for 4 hops, the execution time rose 46.1 seconds when varied from 1 to 3. For 2 and 3 hops, the execution time suffered little variation though rose with 2 hops and dropped with 3.

5.3.1 Maximum Number of Hops Variation: Behavior Analysis

As observed in the results obtained for the Florianópolis instance, as the number of hops increased, the required number of DAPs in the solution decreased but slower. It is not always true, however, that the number of DAPs will decrease every time the maximum number of hops is incremented. Eventually, a constant value is reached, indicating that allowing more hops will not improve the solution's quality further more. In an extreme scenario, this constant value is equal to 1. Nonetheless, the constant value and the number of hops needed to reach it are linked to the position of smart meters in the instance and their communication range. Figure 5.8 exhibits the solution's quality for all 4 instances to a maximum of 20 hops considering a redundancy of 1.

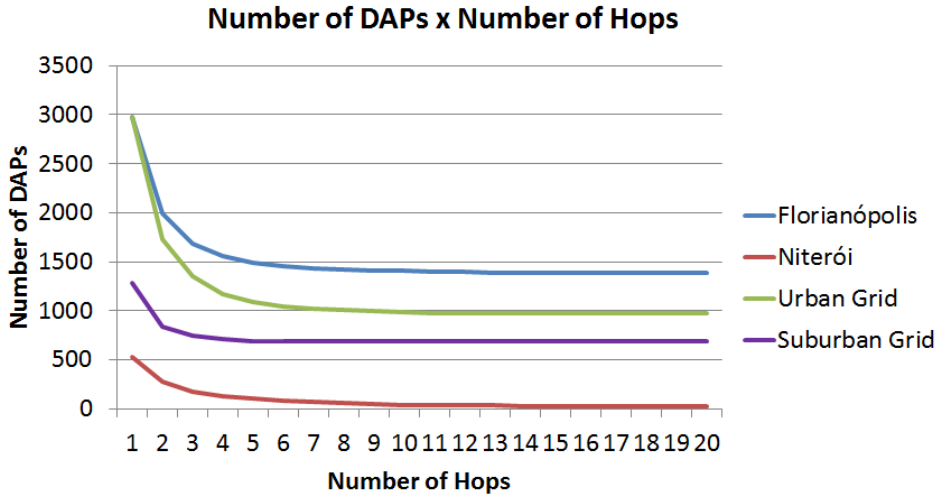
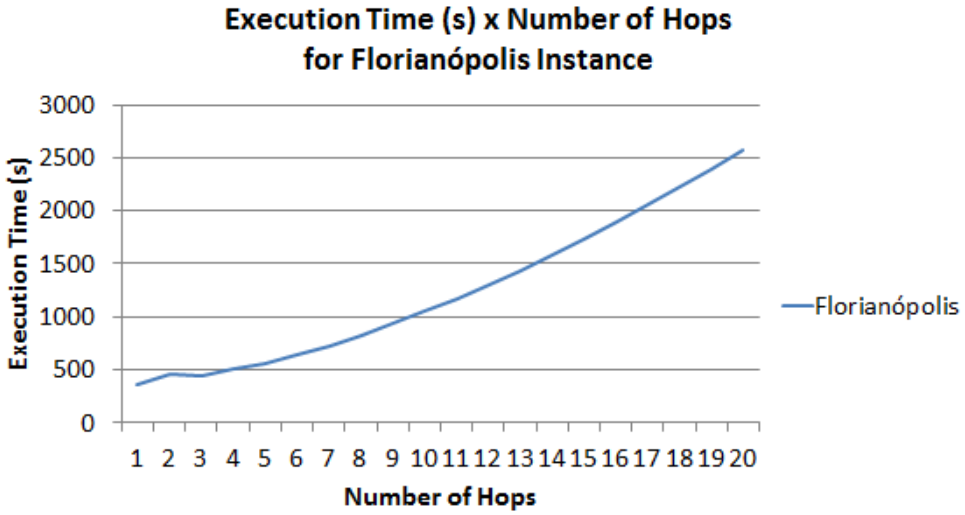
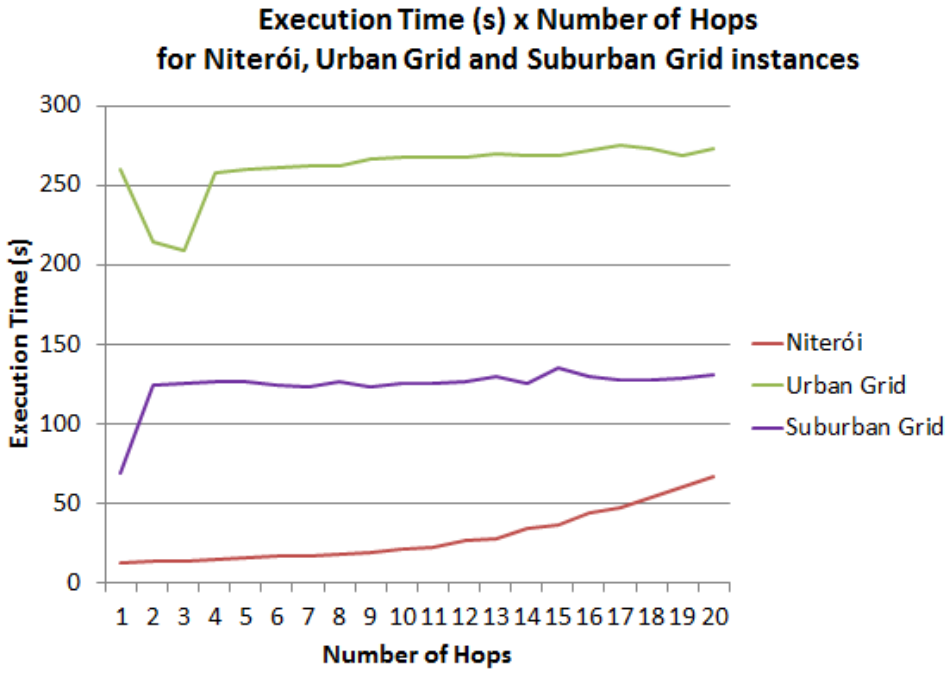


Figure 5.8: Solution's quality with varying number of hops considering a redundancy value of 1.

The graphic shows that all instances behave in the same way, with the number of DAPs quickly dropping at the start to finally reach a constant value.



(a) Execution time in seconds per hop for the Florianópolis instance.



(b) Execution time in seconds per hop for Niterói, Urban Grid and Suburban Grid instances.

Figure 5.9: Execution time in seconds per hop for all instances considering a redundancy value of 1.

Figure 5.9 shows the execution time used to obtain the solution for each hop limit. The graphic for Florianópolis instance is separated for a better visualization. For both Florianópolis and Niterói instances, the execution time rose as the number of hops increased. This indicates that new coverage choices kept being created by the hop increment, which consumes time when creating the coverage matrix. However, for the Urban and Suburban

grid instances the execution time was maintained similar for all hops with little variation, indicating that though the number of hops have been increased, it had little or no impact on the instances' densities. This little impact is associated to the instance's characteristics such as meter and installation site positions, meaning that the hop increment was unable to create new communication paths and, for this reason, did not impact on the execution time. Indeed, this relation can be proven by the graphics in Figures 5.10 and 5.11.

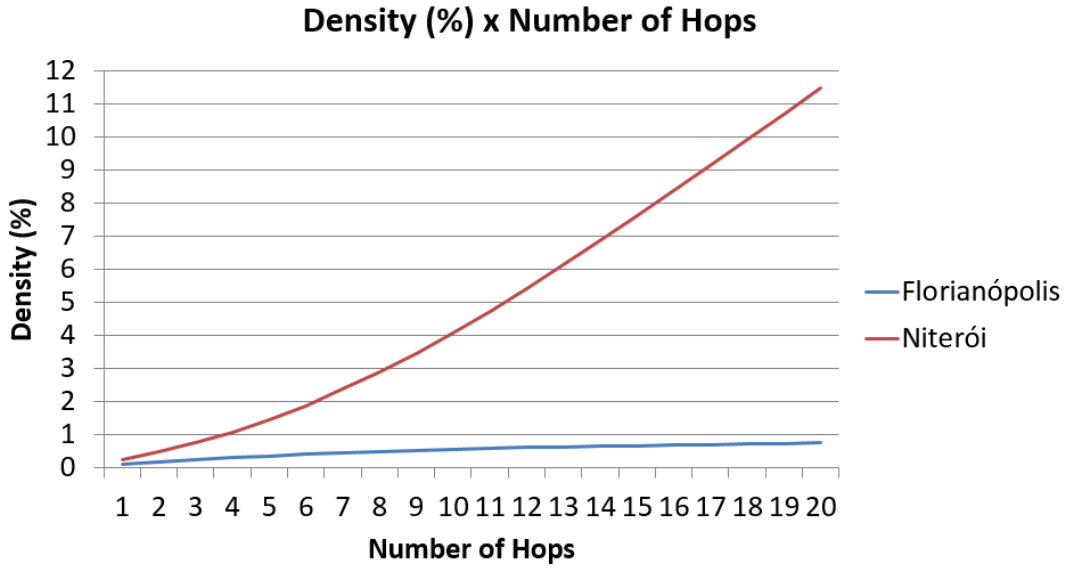


Figure 5.10: Instance density per hop for the Florianópolis and Niterói instances.

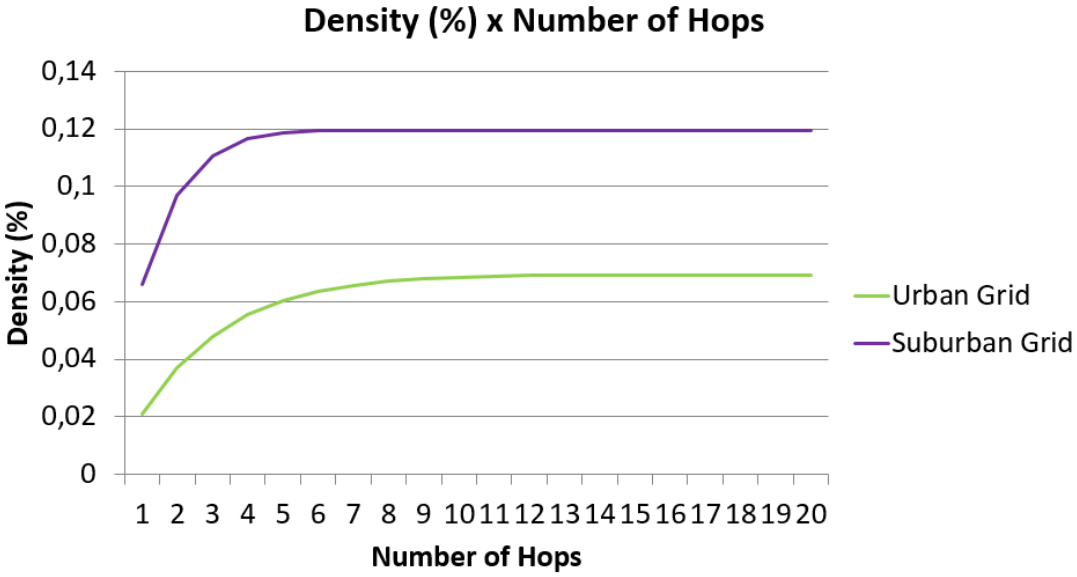


Figure 5.11: Instance density per hop for the Urban and Suburban Grid instances.

For Urban and Suburban Grid instances the density variation was negligible enough not to impact on the execution time and remained constant after 12 hops for the Urban Grid and after 8 hops for the Suburban Grid. On the other hand, for Florianópolis and

Niterói, the density kept growing. Additionally, the density growth by incrementing the number of hops for these instances is, in absolute value, much higher than those of both Urban and Suburban Grid instances, which is why the execution time growth became more evident. It is also important to notice that, although the density of Florianópolis did not rise as fast as Niterói's, it had a greater impact on it's execution time. This happened because the execution time is also related to an instance's size. A density growth in the instance of Florianópolis corresponds to much more connections (i.e. number of 1's in the coverage matrix) and the time to create the coverage matrix is also greater, since there are more meters and installation sites to analyze.

5.3.2 Redundancy Variation: Behavior Analysis

We conducted an evaluation similar to the previous one but varying the input redundancy value instead of the maximum number of hops. For these results we fixed the maximum number of hops in only 1 unit. In Figure 5.12 we can notice that the number of DAPs in the solutions rapidly grows since the initial redundancy variation. This growth tends to reduce as the redundancy increases until reaching a stabilization value which is no longer able to increase. In the worst case scenario this value is equal to the number of installation sites available. For Niterói, Urban Grid and Suburban Grid instances the stabilization value was found for a input redundancy of 4, 6 and 6 respectively. Additionally, the stabilization value for the Niterói instance corresponds to it's maximum possible value (1030). For Florianópolis, the number of DAPs kept growing for all the 20 redundancy values considered.

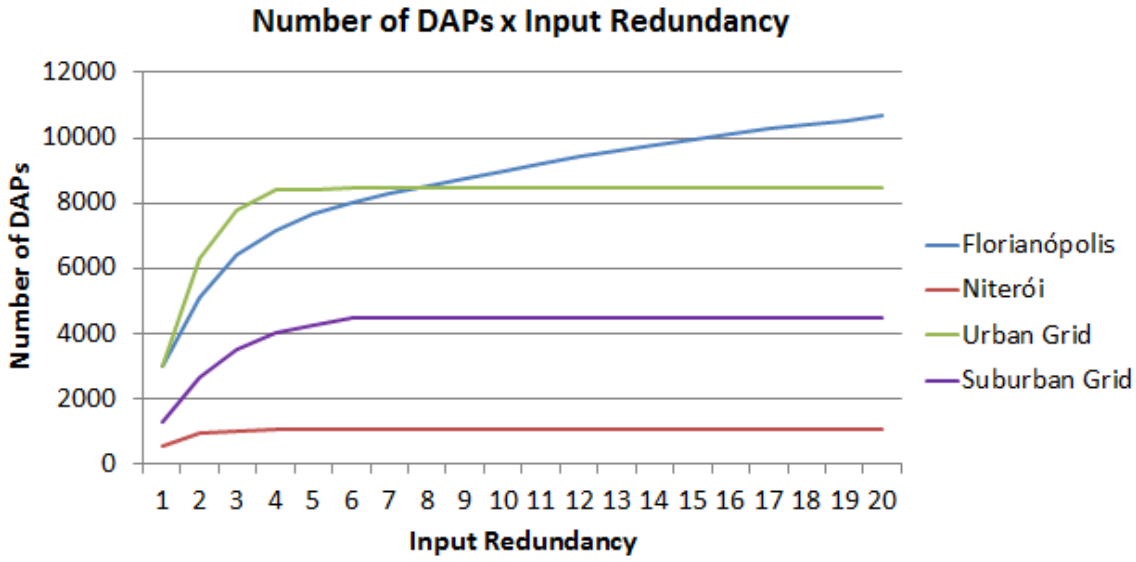


Figure 5.12: Solution's quality with varying redundancy.

The number of DAPs in the solution is also related to each instance's peculiarity. In Figure 5.13, the graphic shows the average redundancy of all meters. For Niterói and Urban Grid the maximum average redundancy was approximately of 2.5 units. This indicates that, due to the position of meters and installation sites, an instance may possess a limitation on providing redundancy coverage for some of its meters and increasing even more the maximum input redundancy does not affect the solution. For the Suburban Grid instance, the maximum average redundancy was of approximately 3.25, whereas for Florianópolis, the average redundancy kept growing, as expected. In some cases, the average redundancy in Florianópolis was greater than the associated input redundancy, this can be explained because the optimal position of DAPs to ensure the redundancy restriction for some meters end up causing the redundancy of other meters to be increased even further. Therefore, it is possible that when the target redundancy is set to 2, for example, some meters may even be covered by much more DAPs, which increases the average redundancy.

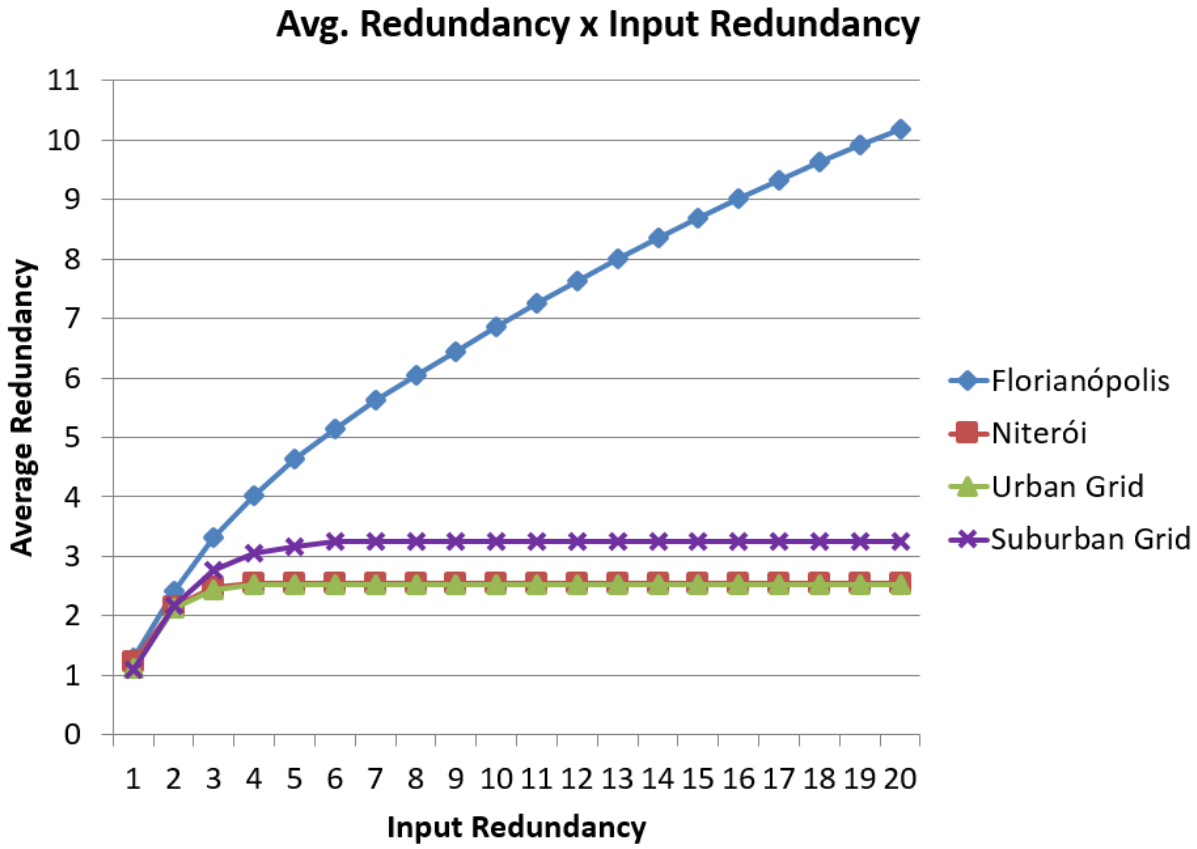


Figure 5.13: Instance's average redundancy for each input redundancy.

Finally, Figure 5.14 compares the execution time for all instances. There is no visible impact on the execution time caused by the redundancy variation, though for the Florianópolis instance the execution time fluctuated from 680 s to 800 s approximately.

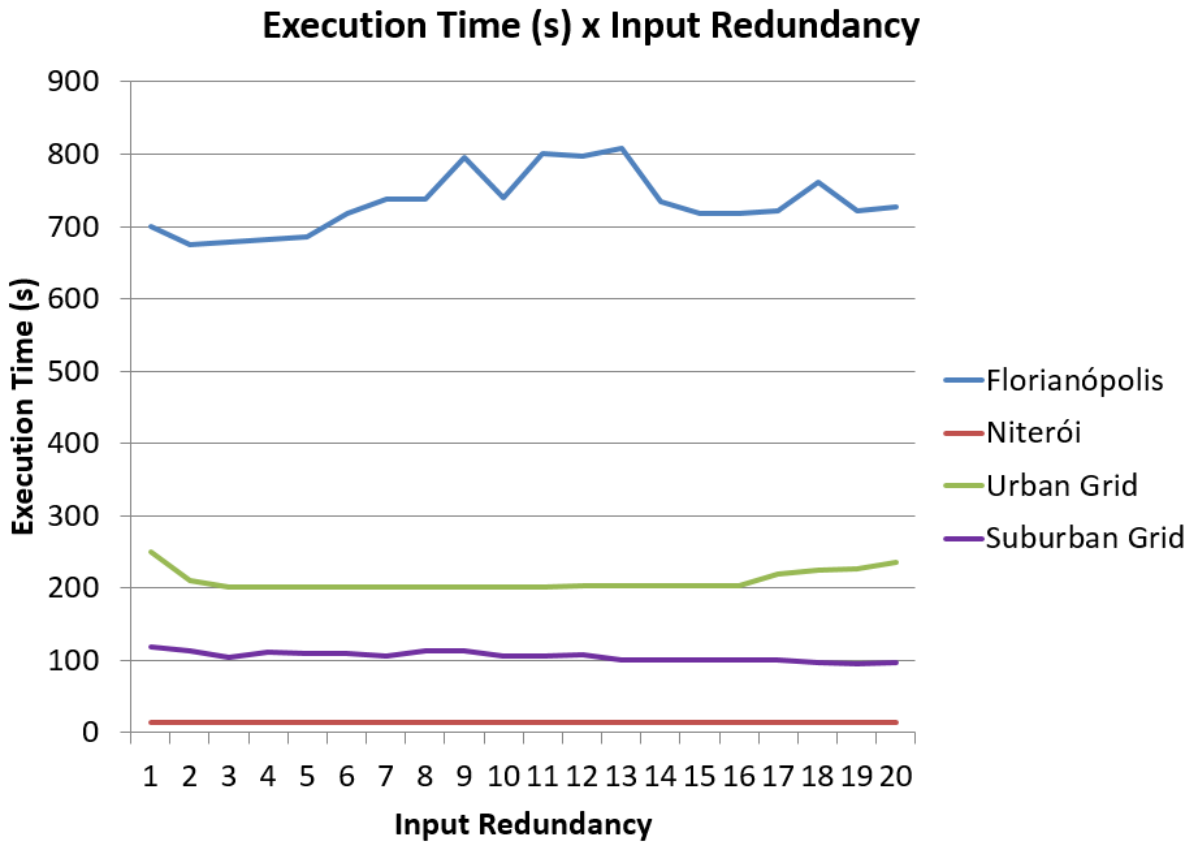


Figure 5.14: Execution time in seconds for varying redundancy.

5.4 Memory Limit Variation Experiment

Since the optimal solution for the Florianópolis instance is missing, we cannot determine how close to the optimal the MOSKOU heuristic's solution is. The objective of the experiment detailed in this section is to analyze the effectiveness of the MOSKOU split by varying the maximum RAM memory limit. We expect that the heuristic is able to find solutions equal or near to the optimal even when the instances are divided in multiple sub-instances. Therefore, it is possible to infer that for big instances, where the optimal solution is missing, the MOSKOU heuristic is capable of obtaining good solutions. To this intent, we gradually varied the memory limit in steps of 10 MB of RAM and collected the solution's quality for each step for the Niterói instance. This experiment was run for the Niterói instance because its optimal solutions is known and is the instance which more closely resembles the Florianópolis one.

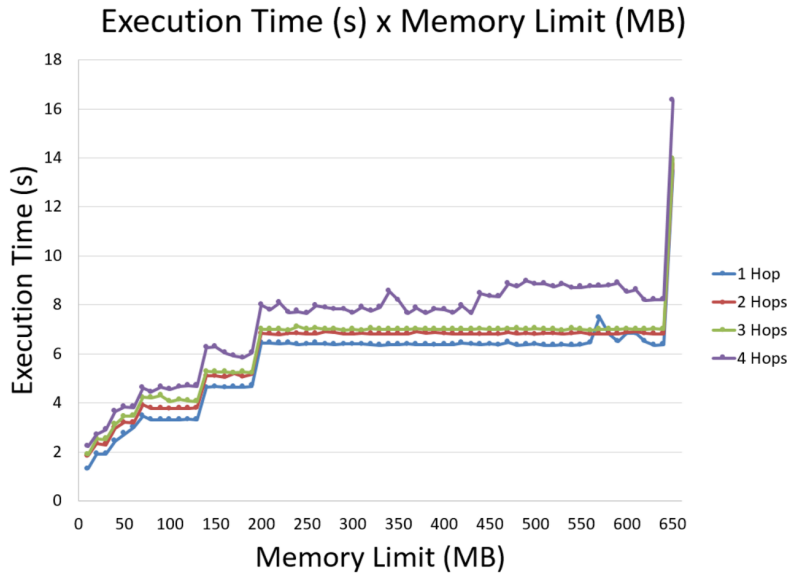
Figure 5.15 compares the solution's gap (with respect to the optimal solution) and execution time per memory limit value. We display results for varying number of maximum hops from 1 to 4 and fixed redundancy value of 1. The lesser the memory limit, the

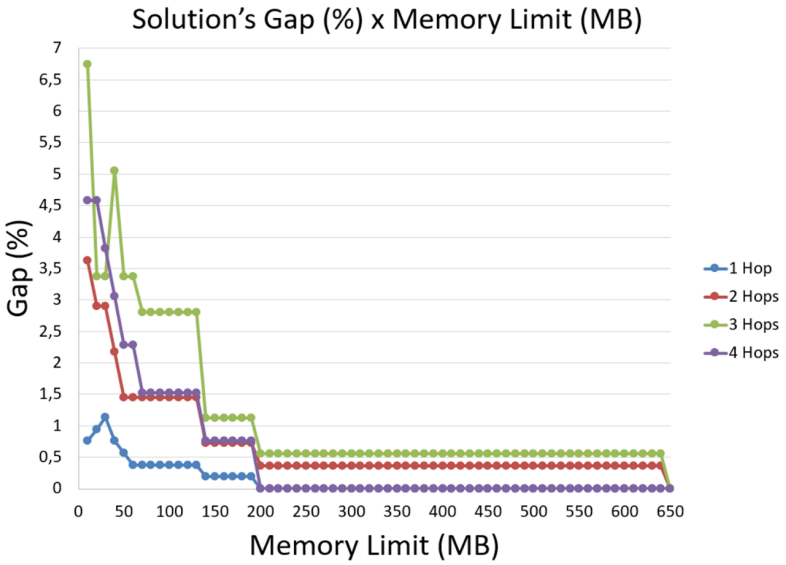
greater the number of sub-instances generated. The number of sub-instances per memory limit value is presented at the bottom of the figure for each hop limit. For the Niterói instance, with 650 or more megabytes of RAM the MOSKOU does not perform splits since it is able to solve it optimally.

Notice that even with very restricted amounts of memory limitation the MOSKOU heuristic is capable of finding near-optimal solutions. The worst obtained solution occurred for 3 hops with a gap of 6.74%. However, for the same value of memory limit, the execution time was reduced by 7.4 times. For 1 and 4 hops, the MOSKOU heuristic was able to find the optimal solution with 200 MB of RAM by merging the solutions of 2 sub-instances. On the other hand, for 2 and 3 hops, the MOSKOU heuristic could only obtain the optimal solution without splitting the original instance.

The graphics also show a strange behavior for 1 and 3 hops, in which a worse solution was found when the memory limit increased and the available memory was very scarce. We can see that with 1 hop for this instance, the solution obtained with 10 MB was better than those with 20 and 30 MB. Analogously, with 3 hops, the solution worsened when increasing the limit from 30 to 40 MB. This occurrence can be explained by the post-optimization method that is used in the MOSKOU heuristic. For these values, the instance was divided in different number of sub-instances, and when the post-optimization method tried to reduce the number of DAPs, it ended having a better solution by optimizing the sub-instances generated by the smaller memory limit. However, this event only became evident due to the very small amount of memory considered (10 - 40 MB) which forced the instance to be split multiple times. We believe that this problem was induced due to the very restricted amount of memory used because it only occurred with less than 40 MB. It is also important to notice that even with 10 MB memory limit, the worst solution gap obtained was less than 7%.

The graphics from Figure 5.15 allow us to conclude that, in general, as the memory limit grows, so does the solution's quality and the execution time. Naturally, it is recommended to use the maximum amount of RAM memory available. Furthermore, these results indicate that the divide-and-conquer approach is suitable for Smart Grid instances. Additionally, whenever an instance is large enough to fail to achieve its optimal solution given the available memory, the MOSKOU heuristic is able to return a good quality solution based on the user's chosen memory limit value.





For 1, 2 and 3 hops

Memory Range (MB)	Number of sub-instances
10	12
20-30	8
40	6
50-60	5
70-130	4
140-190	3
200-640	2
> 650	1

For 4 hops

Memory Range (MB)	Number of sub-instances
10	14
20	9
30	8
40	6
50-60	5
70-130	4
140-190	3
200-640	2
>650	1

Figure 5.15: Execution time and solution's gap with varying memory limit. The number of sub-instances generated is also presented for each hop.

5.4.1 Solving Time Limit Evaluation

As mentioned in the beginning of this chapter, a solving time limit of 60 seconds was established for the GLPK solver. In other words, after the problem is created and the memory allocated in the solver, the optimal solution is to be found in a maximum of 60 seconds. If the optimal solution is not found until then, the solver returns the best solution found so far. It has been observed during the development of this work that if most optimal solutions are not found in few seconds, the solver is not able to find it in a reasonable amount of time. On the other hand, it has also been observed that the solutions found by the solver until the time limit is reached are very close to the optimal (since the solver shows the maximum gap for each found solution). In Figures 5.16 and 5.17 the maximum gap values are presented for both Urban and Suburban Grid instances respectively. Because the MOSKOU heuristic divided the Urban Grid in 2 sub-instances, the gap corresponds to the average of both sub-instances.

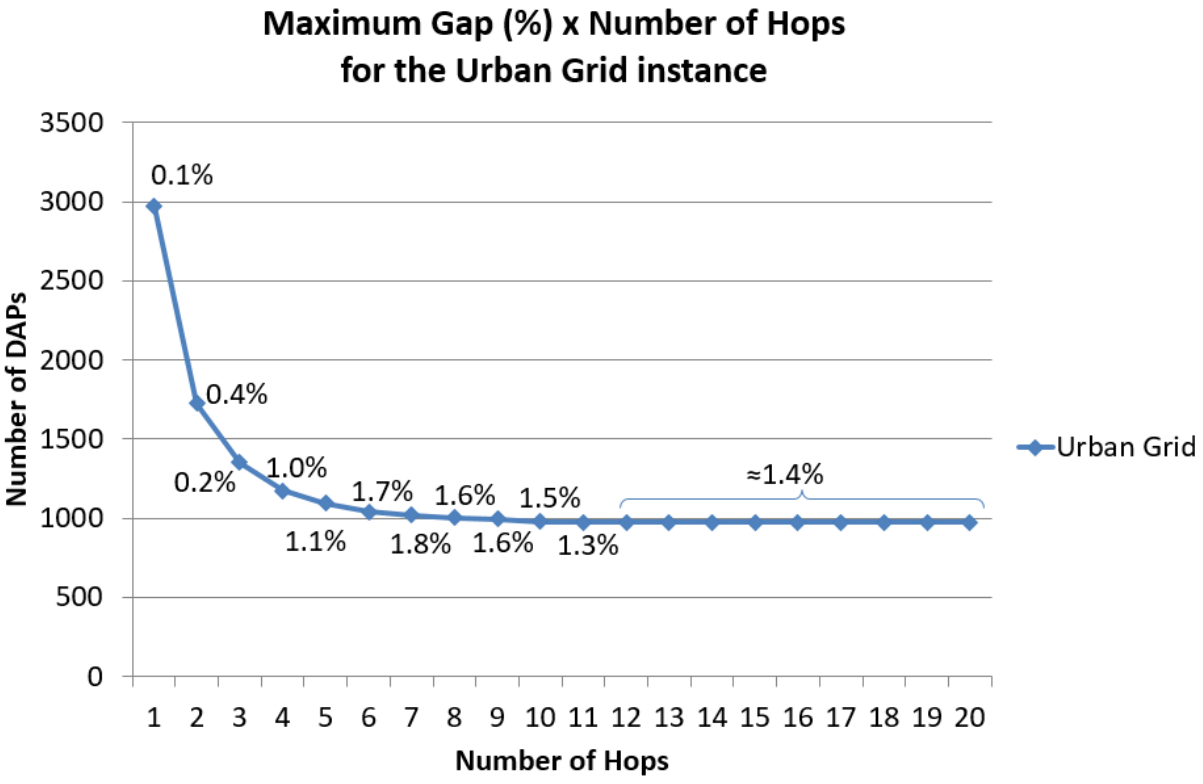


Figure 5.16: Solution and maximum gap from the optimal for the Urban Grid instance.

As both graphics show, with 1 minute limit the maximum obtained gap for both instances was only of 1.8%. Furthermore, for the Suburban Grid instance with 1 hop, the time limit was not exceeded and the optimal solution was found. It is expected that with more hops, the additional coverage information complicates the problem and, for

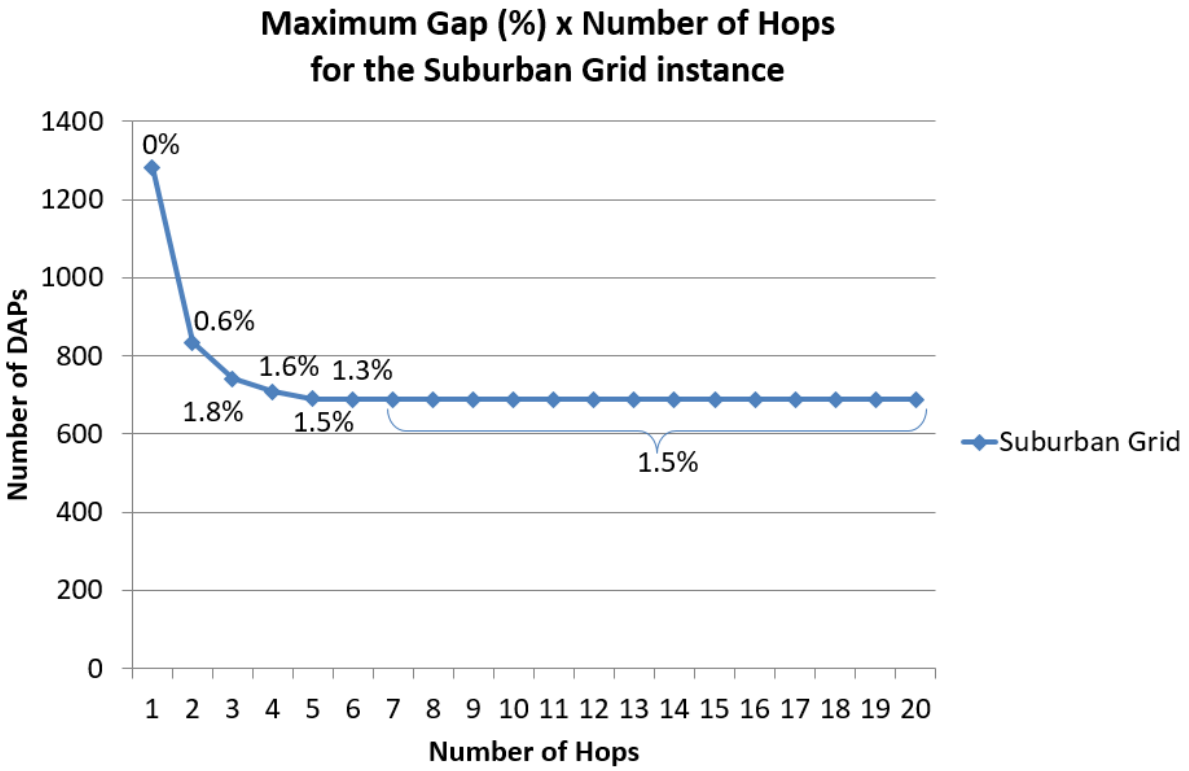


Figure 5.17: Solution and maximum gap from the optimal for the Suburban Grid instance.

that reason, the time limit ends up being reached. Though it has been observed that the smallest gaps were obtained with small amount of hops (1 to 3), specially for the Urban Grid instance, the results do not allow us to conclude that the gap increases with the number of hops (i.e. density).

Chapter 6

Conclusion

This work presented a reduction of the DAP planning problem in Smart Grids to a well-known optimization problem and proposed a heuristic method to solve it. We also extended the problem providing a new mathematical formulation to consider a redundancy based planning, which is essential to maintain network robustness, softening the impact of a DAP failure. Our proposed heuristic, named MOSKOU, employs a divide and conquer approach that splits an instance based on the estimated amount of RAM memory, which can be calculated based on an instance's dimension. A memory based split was utilized because results have shown that the SCP created based on Smart Grids is rapidly solved but consumes a huge amount of memory due to its dimension. The split is based on the K-Means algorithm which groups meters and installation sites that are geographically related to one another. Each generated sub-instance is independently applied to a solver, the solutions merged and applied to a post-optimization method that softens the inaccuracy imposed by the split. This divide-and-conquer approach is ideal for the AMI planning because far regions do not interfere with each other due to the limitations of a DAP range. In this way, it is possible to find a near-optimal solution by dividing a city, for example, in regions that are independent from each other, such as neighborhoods.

The MOSKOU heuristic was compared to a greedy approach and to the Grid heuristic, which was developed on the early stages of this work. Results have shown that although the greedy heuristic is able to find solutions in just a few seconds, their solutions are poor and far from the optimal. The only difference between the Grid and MOSKOU heuristics is the split method, whenever there is not enough memory to optimally solve an instance. The K-Means split method used by the MOSKOU heuristic is able to divide the instance in groups that include meters and installation sites that are geographically related to one another. Moreover, this approach allows the instance to be split in less sub-instances in

comparison to the Grid heuristic. Consequently, the MOSKOU heuristic's execution time is reduced and solution's quality improved in comparison to the Grid heuristic.

It is important to remember that our model receives as input the type of scenario (urban, suburban or rural), communication technology, devices transmission power, maximum number of hops and desired redundancy. These input values must be consistent with the instance in which the DAPs will be deployed. If we poorly characterize an instance's scenarios, considering an urban center as rural for example, even if the optimal solution is found, it will not have any usefulness in a real deployment. The same applies when determining the maximum number of hops: allowing high values of hop limit reduces the amount of DAPs on the solution but may consider poor or even non-existent communication paths on a real scenario, specially in urban regions that contain high sources of noises and interference.

6.1 Contributions

The MOSKOU heuristic is one of the functionalities available in the TELE-SIRIS (Módulo de Telecomunicações de um Simulador de Redes Inteligentes) project developed in association with Neo Domino and ANEEL (Agência Nacional de Energia Elétrica). With the TELE-SIRIS module, the user is able to deploy DAPs over a map and visualize the connections between devices in real time. All input values are adjustable in the user interface, such as the scenario, technology, transmission power, maximum number of hops and redundancy. The module also allows the user to save the current planning and load it. In Figure 6.1 a planning example is displayed in the TELE-SIRIS module. Other functionalities are also available, such as visualizing 3G/4G/GPRS DAP coverage area and collecting planning statistics.

The following publications are results of this work and the TELE-SIRIS module.

1. Rolim, G., Passos, D., Moraes, I., & Albuquerque, C. (2015, October). **Modelling the Data Aggregator Positioning Problem in Smart Grids**. In Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on (pp. 632-639). IEEE.

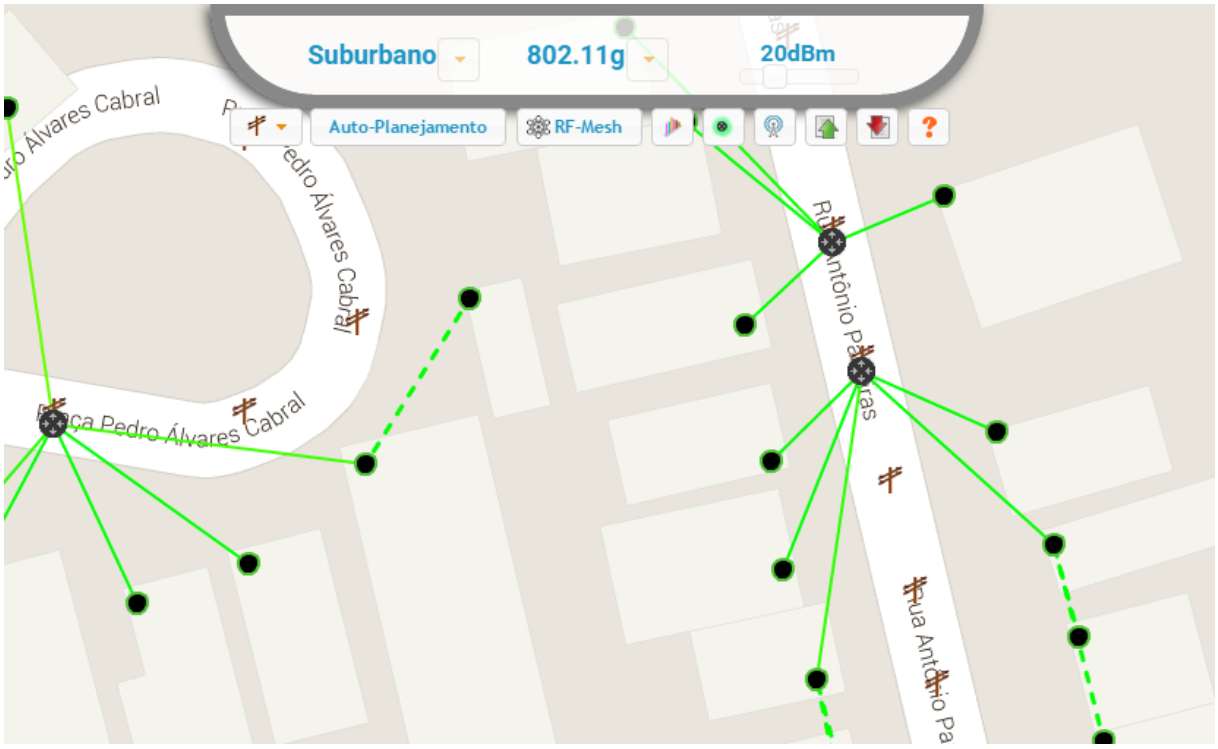


Figure 6.1: TELE-SIRIS module's screen.

2. Rolim, G., Moraes, I., & Albuquerque, C. (2015, May). **Modelo e Solução para o Problema de Posicionamento de Agregadores em Redes Elétricas Inteligentes**. XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos.

3. Rolim, G., Moraes, I., & Albuquerque, C. (2015, May). **Smart Planner: Uma Ferramenta de Planejamento para Smart Grids**. Salão de Ferramentas - XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos.

4. Rolim, G., Silva, P., Albuquerque, C., Carrano, R., Moraes, I., Bettiol, A., Carniato, A., Passos, L., Homma, R., Andrade, R., Molina, F., Kinceler, R. & Filho, S. I. V. (2015, August). **Sistema para Planejamento de Instalação de Agregadores em Redes Elétricas Inteligentes**. Congresso de Inovação Tecnológica em Energia Elétrica 2015 - CITENEL 2015.

5. Rolim, G., Sousa, C., Carrano, C., Moraes, I., Albuquerque, C., Bettiol, A., Carniato, A., Passos, L., Homma, R., Andrade R. & Molina, F. (2015, June). **Smart Grid Deployment Planning: Case Study Covering a Brazilian Feeder in Automation Process**. 23rd International Conference on Electricity Distribution - CIRED

6. Rolim, G., Albuquerque, C., Moraes, I., Bettiol, A., Carniato, A., Passos, L., Homma, R. & Molina, F. (2015, May). **Sistema de Posicionamento de Agregadores de Dados em Redes Elétricas Inteligentes**. XVI ERIAC Encuentro Regional Iberoamericano de Cigré

7. Sousa, C., Rolim, G., Carrano, C., Albuquerque, C., Bettiol, A., Carniato, A., Passos, L., Homma, R., Andrade R. & Molina, F. (2015, October). **Link Quality Estimation for Advanced Metering Infrastructure**. Innovative Smart Grid Technologies Conference Latin America - ISGT-LA.

8. Rolim, G., Passos, D., Albuquerque, C., Moraes, I., Carrano, R., Sousa, C., Bettiol, A., Passos, L., Homma, R., Andrade, R. & Molina, F. (2016, May). **Scalability Evaluation of the Data Aggregator Positioning Problem in Smart Grids**. IEEE/PES T&D Latin America, 2016.

6.2 Future Work

For future work, it is essential that our planning results are verified in a real deployment environment. We expect that our conservative approach for determining good quality links is able to well represent the environment even though all sources of noises and interferences are nearly impossible to determine. Moreover, as mentioned in Chapter 5, we recommend for future work a modification of the MOSKOU heuristic that allows it to use a distributed solver running on multiple machines. In this way, the heuristic will work with a greater amount of RAM and obtain better solutions. Finally, we also recommend for future works an adaptation of our method to consider a new input that determines a maximum amount of money that must be spent in the planning and considers different types of DAPs, each with its own hardware peculiarities and buying price. Therefore, the heuristic must also take in consideration what type of DAP to deploy on an installation site

References

[1] NIST PAP2 Guidelines for assessing wireless standards for smart grid application., 2012.

[2] Aalamifar, F.; Shirazi, G. N.; Noori, M.; Lampe, L. Cost-efficient data aggregation point placement for advanced metering infrastructure. In *IEEE International Conference on Smart Grid Communications (SmartGridComm), 2014* (2014), pp. 344–349.

[3] CEPT Administrations. Monte-carlo simulation methodology for the use in sharing and compatibility studies between different radio services or systems. *ERC within the CEPT* (2000).

[4] Cheng, P.; Wang, L.; Zhen, B.; Wang, S. Feasibility study of applying lte to smart grid. In *2011 IEEE First International Workshop on Smart Grid Modeling and Simulation (SGMS)* (2011), IEEE, pp. 108–113.

[5] Damosso, E.; Correia, L. M. *COST Action 231: Digital Mobile Radio Towards Future Generation Systems: Final Report*. European Commission, 1999.

[6] Extended Hata SRD for short range devices implemented in the SEAMCAT simulator. `http://tractool.seamcat.org/wiki/Manual/PropagationModels/ExtendedHata`. Accessed on February, 2015.

[7] Fang, X.; Misra, S.; Xue, G.; Yang, D. Smart grid the new and improved power grid: A survey. *Communications Surveys & Tutorials, IEEE 14*, 4 (2012), 944–980.

[8] Farahani, R. Z.; Asgari, N.; Heidari, N.; Hosseininia, M.; Goh, M. Covering problems in facility location: A review. *Computers & Industrial Engineering 62*, 1 (2012), 368–407.

[9] Farhangi, H. The path of the smart grid. *IEEE Power and Energy Magazine 8*, 1 (2010), 18–28.

[10] Galli, S.; Scaglione, A.; Wang, Z. Power line communications and the smart grid. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on* (2010), IEEE, pp. 303–308.

[11] Gungor, V. C.; Lu, B.; Hancke, G. P. Opportunities and challenges of wireless sensor networks in smart grid. *IEEE Transactions on Industrial Electronics 57*, 10 (2010), 3557–3564.

[12] Gungor, V. C.; Sahin, D.; Kocak, T.; Ergut, S.; Buccella, C.; Cecati, C.; Hancke, G. P. A survey on smart grid potential applications and communication requirements. *IEEE Transactions on Industrial Informatics 9*, 1 (2013), 28–42.

[13] Hatay, M. Empirical formula for propagation loss in land mobile radio services. *IEEE Transactions on Vehicular Technology 29*, 3 (1980), 317–325.

[14] IBM. CPLEX optimizer: High-performance mathematical programming solver for linear programming, mixed integer programming, and quadratic programming.

[15] Karp, R. M. *Reducibility among combinatorial problems.* Springer, 1972.

[16] Lai, D.; Manjeshwar, A.; Herrmann, F.; Uysal-Biyikoglu, E.; Keshavarzian, A. Measurement and characterization of link quality metrics in energy constrained wireless sensor networks. In *IEEE GLOBECOM'03* (2003), vol. 1, pp. 446–452.

[17] Ma, R.; Chen, H.-H.; Huang, Y.-R.; Meng, W. Smart grid communication: Its challenges and opportunities. *IEEE Transactions on Smart Grid 4*, 1 (2013), 36–46.

[18] Makhorin, A. GLPK (GNU linear programming kit), 2008.

[19] Okumura, Y.; Ohmori, E.; Kawano, T.; Fukuda, K. Field strength and its variability in VHF and UHF land-mobile radio service. *Rev. Elec. Commun. Lab 16*, 9 (1968), 825–73.

[20] Parikh, P. P.; Kanabar, M. G.; Sidhu, T. S. Opportunities and challenges of wireless communication technologies for smart grid applications. In *Power and Energy Society General Meeting, 2010 IEEE* (2010), IEEE, pp. 1–7.

[21] Samadi, P.; Mohsenian-Rad, A.-H.; Schober, R.; Wong, V. W.; Jatskevich, J. Optimal real-time pricing algorithm based on utility maximization for smart grid. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on* (2010), IEEE, pp. 415–420.

[22] Sauter, T.; Lobashov, M. End-to-end communication architecture for smart grids. *IEEE Transactions on Industrial Electronics 58*, 4 (2011), 1218–1228.

[23] Souza, G.; Vieira, F.; Lima, C.; Junior, G.; Castro, M.; Araujo, S. Optimal positioning of GPRS concentrators for minimizing node hops in smart grids considering routing in mesh networks. In *IEEE PES Conference On Innovative Smart Grid Technologies Latin America (ISGT LA)* (2013), pp. 1–7.

[24] Tomazic, S. *Encyclopedia of Wireless and Mobile Communications.* Taylor & Francis, 2008.

[25] Toregas, C.; Swain, R.; ReVelle, C.; Bergman, L. The location of emergency service facilities. *Operations Research 19*, 6 (1971), 1363–1373.

[26] Usman, A.; Shami, S. H. Evolution of communication technologies for smart grid applications. *Renewable and Sustainable Energy Reviews 19* (2013), 191–199.

[27] Walfisch, J.; Bertoni, H. L. A theoretical model of uhf propagation in urban environments. *IEEE Transactions on Antennas and Propagation 36*, 12 (1988), 1788–1796.

[28] Wenpeng, L. Advanced metering infrastructure. *Southern Power System Technology 3*, 2 (2009), 6–10.

[29] Yarali, A.; Rahman, S. Smart grid networks: Promises and challenges. *Journal of Communications 7*, 6 (2012), 409–417.

[30] Yelbay, B.; Birbil, S. I.; Bülbül, K. The set covering problem revisited: An empirical study of the value of dual information. *MANAGEMENT 11*, 2 (2015), 575–594.

[31] Zhou, J.; Hu, R. Q.; Qian, Y. Scalable distributed communication architectures to support advanced metering infrastructure in smart grid. *IEEE Transactions on Parallel and Distributed Systems 23*, 9 (2012), 1632–1642.

APPENDIX A – Density Based Creation Algorithm

```
   input  : meters, poles, density
   output: coverage_matrix
   // density value is a number between 0(0%) and 1(100%)
1  num_of_1s ← (|meters| × |poles|) × density;
2  i ← 0;
3  meters_aux ← copy of meters;
4  while num_of_1s > 0 do
5  |  to_be_covered ← choose_meter(meters_aux);
6  |  meters_aux ← meters - to_be_covered;
7  |  coverage_matrix ← poles[i] now covers to_be_covered;
8  |  i++;
9  |  num_of_1s−;
10 |  if |meters_aux| == 0 then
11 |  |  meters_aux ← copy of meters;
12 |  end
13 |  if i == |poles| then
14 |  |  i ← 0;
15 |  end
16 end
17 return coverage_matrix;
```

Algorithm 4: Algorithm to create instances based on density.

The first step of this algorithm is to determine the number of pairs that must be set to 1. This number is represented by the variable num_of_1s in line 1 and is obtained based on the number of meters, poles and density. Line 2 initializes a variable i which represents the pole that will cover the chosen meter. $meters\_aux$ starts as a copy of the $meters$ set. While the num_of_1s is greater than 0 (i.e the density has not been reached yet) a random meter is chosen from $meters\_aux$ (line 5). This meter is removed from $meters\_aux$ (line 6) and is now covered by the pole represented by i. i is incremented, indicating that the next pole in line will cover the next chosen meter. If the $meters\_aux$ set is empty (line 10), it becomes the original $meters$ set once again. When i reaches the size of poles it is reset to 0, restarting the sequence of poles that will cover the chosen meter. When the algorithm ends, the created coverage matrix is returned.

APPENDIX B – Grid Heuristic Algorithm

Like MOSKOU, the Grid heuristic employs a divide-and-conquer approach but, instead of using K-Means, the problem is divided in fixed sized cells which are independently applied to the exact method. Each cell is a square with size obtained according to the memory estimation calculation so that no cell bursts the memory limit. For each cell, an SCP is created from the meters and poles of this cell plus the poles of the neighbor cells. The poles from neighbor cells are utilized to increase the degree of choice during the execution. The union of all cells solutions compose the solution to the initial complete problem. This solution is then submitted to a post-optimization method for refinement. Figure B.1 shows the heuristic division procedure. The initial problem, shown as P1, is divided into a grid G. For each cell of this grid, a sub-problem is created with the meters and poles of a cell plus the poles of the neighbor cells. The cell C1 generates the sub-problem S1, C2 generates the sub-problem S2 and C3 generates the S3. Each sub-problem is optimally solved. The union of the results for S1, S2 and S3 composes the solution to the problem P1.

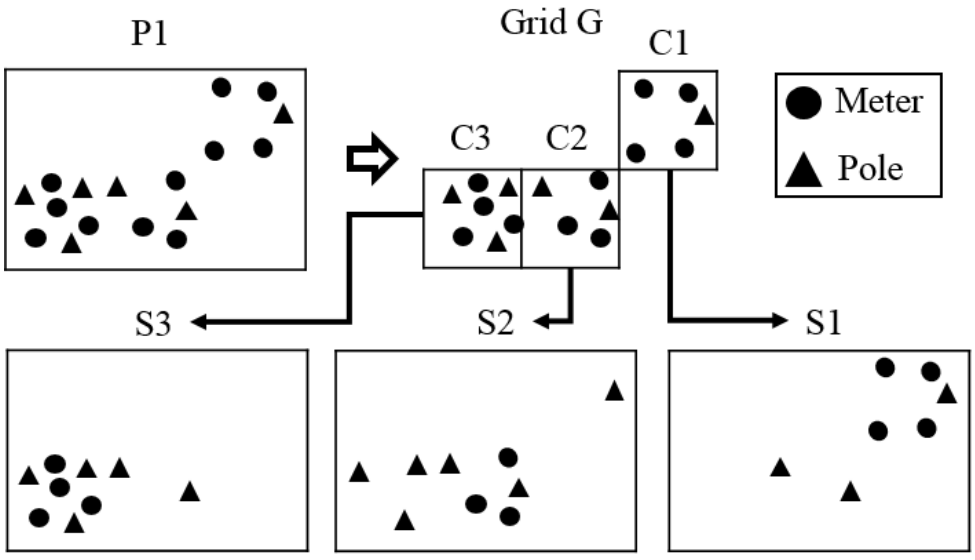


Figure B.1: Division process of the proposed Heuristic.

APPENDIX C – BER Calculation

input : sce, tech, distance, power
output: BER estimation
1 loss ← getAdjustedExtendedHataSRD(sce,tech,distance);
2 ReceiverPower ← power - loss;
3 SNR ← ReceiverPower - getNoise(sce);
4 γb ← SNR/spectral_efficiency;
5 BER← $\alpha Q(\sqrt{\gamma b})$;
6 return BER;

Algorithm 5: BER Calculation.

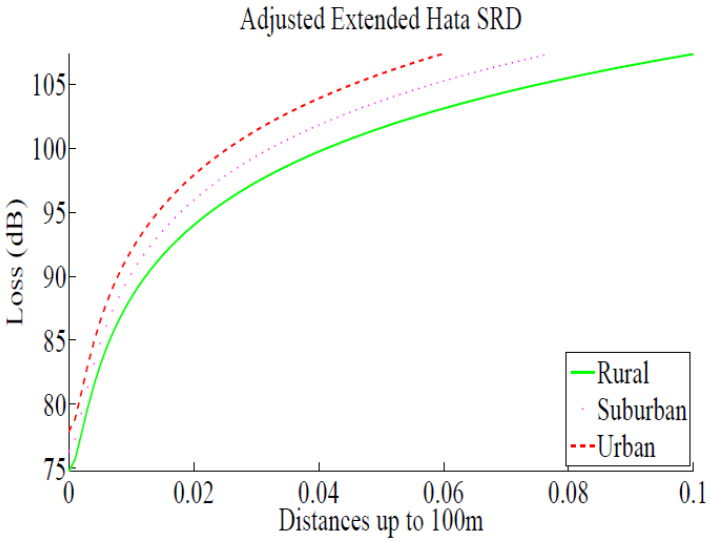


Figure C.1: Loss x Distance in the adjusted curve for the Extended Hata SRD.

Table C.1: Noise values for each scenario.

	Noise Value (dBm)
Urban	-79
Suburban	-81
Rural	-85

Algorithm 5 shows how the BER is calculated. Notice that both $h1$ and $h2$ are ignored because the height difference between a smart meter and a DAP is low enough to

not interfere in the result. The *loss* variable is obtained from the adjusted curves of the Extended Hata SRD propagation model. The adjusted curves relate the distance with a loss value for all three types of scenario. Figure C.1 shows the loss values for all three types of scenarios for 2.4 GHz. This adjusted curve is better explained on the following section. With the estimated *loss*, it's possible to obtain the transmission power at the receiver device, which is calculated as the transmitter *power* minus *loss*. The SNR (Signal to Noise Ratio) corresponds to the estimated transmission power at the receiver minus the noise of the chosen scenario. The used noise values of each scenario are displayed on Table C.1. The SNR per bit (γb) is calculated by normalizing the SNR by the spectral efficiency. The spectral efficiency is described as the amount of useful information that can be transmitted over a given bandwidth during a period of time. Finally, BER is calculated from a Q function which is defined as the probability that a Gaussian random variable with mean 0 and variance 1 is bigger than $\sqrt{\gamma b}$. The value of α is associated to the used modulation. For example, 802.11g devices at 6 Mbps (base rate) uses the BPSK modulation technique and α is set to 0.5. Due to the small load generated in the communication between meters and DAPs, its recommended to use the standard's base bit-rate. Small bit-rates tend to increase the probability of success decoding, thus expanding a DAP's coverage range and reducing the number of DAPs needed to plan a region.

C.1 Extended Hata SRD Curve Adjustment

It's virtually impossible to incorporate all features of a real environment in our SDR calculations. For this reason, a conservative approach is adopted to accommodate more challenging environments. For example, recent field tests using IEEE 802.15.4 devices have shown that wireless links, specifically in Smart Grids, present high packet error rate because of numerous interference sources, noises, dynamic topology changes, fading and obstructions [11]. In this work, the Extended Hata SRD model was adjusted to add extra dBs for the path loss to adopt a more pessimistic approach and compensate for the inability to perfectly model a real environment. In this way, we reduce the chance of representing connections between smart meters and DAPs that can not be established due to the amount of noises and obstacles in the environment. Our adjusted Extended Hata SRD (for 802.11g) is presented in Figure C.1.