UNIVERSIDAD
DE MÁLAGA

LENGUAJES Y
CIENCIAS DE LA
COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

TESIS DOCTORAL

Técnicas de optimización para redes de sensores

Autor
Guillermo Molina Arribere

Director
Dr. Enrique Alba Torres

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

Julio de 2010

El Dr. **Enrique Alba Torres**, Catedrático de Universidad del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga,

Certifica

que D. **Guillermo Molina Arribere**, Ingeniero de Telecomunicación por la Universidad de Málaga, ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, bajo su dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulada

Técnicas de optimización para redes de sensores

Revisado el presente trabajo, estimo que puede ser presentado al tribunal que ha de juzgarlo, y autorizo la presentación de esta Tesis Doctoral en la Universidad de Málaga.

En Málaga, Julio de 2010

Firmado: Dr. Enrique Alba Torres
Catedrático de Universidad
Dpto. de Lenguajes y Ciencias de la Computación
Universidad de Málaga

Optimization Techniques for Wireless Sensor Networks

Guillermo Molina

July 2010

Agradecimientos

Una tesis doctoral es un proceso largo y arduo, que no podría llevarse a cabo si no fuera por la ayuda y contribución de muchas personas al esfuerzo y voluntad que le pone uno mismo. Esta sección constituye un pequeño homenaje y reconocimiento a todas esas personas que, directa o indirectamente, han contribuido a que hoy este trabajo sea una realidad. De bien nacidos es el ser agradecido reza el dicho, y pienso hacer honor al mismo, intentando no extenderme en exceso.

Es justo comenzar por el principio, y este no es otro que señalar al máximo "culpable" de que esta tesis exista, la persona que me animó a realizar el doctorado, y que durante todos estos años ha sabido dirigir y supervisar mi trabajo. Obviamente me refiero a mi director de tesis, Enrique Alba.

También me gustaría darles el merecido reconocimiento a mis compañeros miembros de NEO, ese fantástico grupo humano reunido por Enrique. En primer lugar me gustaría mencionar a Francis y a Gabriel, quienes en más de una ocasión me han ayudado con su enorme experiencia y saber hacer a la hora de plantear trabajos y resolver problemas. En segundo lugar, quiero agradecer al "grupo multi-objetivo" formado por Antonio, Paco y Juanjo, por su ayuda cada vez que he intentado hacer mis 'pinitos' dentro de su especialidad, así como por las fructíferas discusiones y colaboraciones ocasionales que he tenido la suerte de disfrutar. Quiero dar una mención especial a José Manuel y Javier Apolloni, por su ayuda con los algoritmos. Por último, darle ánimo a la nueva hornada del laboratorio (Briseida, Jamal, Javier, Pablo, Martín): que sepan que *sí* que hay luz al final del túnel; a Juanjo y José Manuel no hace falta incluiros en este último grupo, porque vosotros ya estáis más que "averiguaos" ;P.

Me gustaría agradecer a Ana su apoyo, y que esté ahí, a las duras y a las maduras. Que me aguante mis cosas; porque sé que hay veces no me dejo ayudar, y aún así me sabe ayudar.

También tengo que agradecer a mi gente de siempre que hayan estado ahí, porque no todo en la vida se resume en trabajo o investigación. Hace falta tener un espacio para refugiarse y recargar las fuerzas. Gracias a Enrique por acordarse de mí a pesar de la distancia, gracias a Quique y a Kata porque sé que puedo contar con vosotros sin importar la situación ni el tiempo que llevemos sin vernos, y gracias a Charli, un tipo peculiar con una habilidad especial para animar a la gente.

Gracias a Adán y a María, dos tesoros que tuve la fortuna de encontrar en el mar de la universidad. Qué suerte tienen Maribel y Yago :).

Y muchas gracias a mucha más gente, porque completáis mi mundo y le dáis color. A Ángela, Antonio, Aure, Dani (futbolista), Dani (akatsuki), Dikra, Gabino, Héctor, Isa, Jesús, Luisda, Pili, Sebas, Vicky, y muchos más. Gracias a Pepe el del Atlas, gracias al cual todavía me mantengo en forma (algo).

Quiero también agradecer a todos mis compañeros de la clase de chino, en especial a mi profesora Qi, que me permite desconectar y mantener mi mente abierta a nuevas experiencias. Gracias a mis tongxuemen: Esperanza, Wong, Antonio, tocayo, Maria Victoria (ambas), etc... y recordando a Miguel Ángel (D.E.P.).

También, y como no podía faltar, quiero agradecer a mi familia que haya estado ahí. A mis padres Juan Miguel y Elena, y a mi hermano Alberto. A mi abuela Josefa, a la que voy a ver menos de lo que se merece. A mamie, que me ha hecho sentirme orgulloso de mis raices. Y a Albert y Jos. Os quiero.

Financiación

Este trabajo de tesis ha sido parcialmente financiado por el Ministerio Español de Educación el Fondo Europeo de Desarrollo Regional(FEDER) a través del contrato TIN2005-08818-C04-01 (Proyecto OPLINK, http://oplink.lcc.uma.es). También ha recibido financiación de los proyectos nacionales MSTAR (TIN2008-06491-C04-01) y andaluz DIRICOM (P07-TIC-03044). Finalmente, el autor de esta tesis ha recibido una beca FPU (AP2005-0914) del Ministerio Español de Educación.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Every once in a while, a new technological tool appears that revolutions the scientific community as it brings new, unseen, and exciting possibilities for researchers. Examples of this are the telescope ($17th$ century), the microscope ($17th$ century), the radio (late $19th$ century), or the radar (during the 40's), among others. In this thesis work, we consider a technology that has been catalogued by *Business Week* as one of the most impacting technological advances for the 21^{st} century: Wireless Sensor Networks (WSNs). The idea behind this technology is not completely novel, since networks of sensors exist since as soon as the 50's. For instance, during the Cold War the US military develop a network of acoustic sensors whose purpose was to detect the presence of soviet submarines approaching the US coast, by establishing a detection frontier in the northern Atlantic ocean (Sound Surveillance System, SOSUS). Later, in the seventies, the DARPA initiated a project to develop networks of nodes for tracking applications with military purposes.

However, the modern concept of Wireless Sensor Networks holds a series of innovative concepts with respect to its predecessors. New networks are designed from a more general purpose perspective, whereas the previous examples were completely application-specific. The stress in new networks is put onto distributed, ubiquitous computing by small and unobtrusive devices. The system must be capable to work autonomously, adaptively to environmental changes, in real time, efficiently, etc. Besides, the range of possibilities and applications is no longer restricted to the military field (although it still remains one of the most important domains), as comfort and economic profit applications begin to take form. The new form of sensor networks is expected to become a technology of wide and general use, just the way mobile telephony or the Internet have become during the last two decades.

Nevertheless, these new and powerful dynamic ad hoc systems also have a set of hard constraints that need to be handled in order to achieved the desired features. This results in new optimization challenges, and unsolved optimization problems, which often happen to be NP-hard, and cannot thus be efficiently solved by classic optimization techniques. These problems integrate novel models, one or more opposing objectives, and constrained resources. These constrained resources are twofold: on the one hand the object of the resolution process, the solution itself, has to handle restrained basic resources (economic, spacial, energetic) and achieve ambitious objectives, and on the other hand the very optimization process is also restricted, since it must be performed in short time on a limited computation platform (the WSN). Additionally, the solution must be used for a long term deployment on a harsh, and unpredictably changing environment. Thus the resolution process has to be swift, without incurring a high computational cost, yet the solution found must be robust and accurate.

There is a kind of optimization technique that has been widely used for complex optimization problems, NP complex problems, which arguably produces near-optimal solutions in a timely fashion: Metaheuristics.

These techniques, which are quickly becoming more and more popular within the research community, are typically iterative processes, which refine a (set of) candidate solution(s). Among the advantages one can find in using metaheuristics are the following:

- Reduced computational complexity.
- Tunable computational effort (trade-off with solution quality).
- Do not require full problem knowledge, just a quality measuring function for any given candidate solution to guide the search.
- Adaptability to handle different paradigms such as multi-objective or distributed optimization.
- Can be tailored (within some range) to fit the computation power, memory size and time limit constraints.

All of these features suggest that Metaheuristics are appropriate techniques to be used to solve the optimization problems in the domain of WSNs. Therefore, we set as the main goal of this thesis work to prove the feasibility of metaheuristic-based resolution techniques in this domain. For this, we identify some important optimization problems found in this domain, propose a set of metaheuristic optimization algorithms to tackle them, and show their effectiveness through statistically assessed experimental evaluation. Our aim when selecting the problems has a strong focus towards applicability of the results obtained; in this sense, we avoid purely academic problems and choose problems that either can be found in real applications or have a clear connection with problems that are currently in need of a solution.

1.2 Objectives and phases

This thesis addresses the resolution of complex optimization problems in the domain of Wireless Sensor Networks through the use of metaheuristic algorithms. This general objective can be detailed into more specific goals as follows:

- Identify the most important problems that arise in the new field of sensor networks. Select a set of problems to be solved in this thesis work.
- Propose a formulation for each of the problems selected.
- Description of the optimization techniques that will be used to solve the problems.
- Propose an application method of the optimization techniques that leads to the resolution of the problems.
- Develop novel tools or techniques that enhance the performance of current optimization techniques, either from the perspective of the quality of the solutions produced, or from the perspective of the computational effort required to reach them; demonstrate their effectiveness through statistically assessed experimental evaluation.

In order to fulfill the thesis objectives, the work has been carried out as follows. We first review the existing research in the field of WSNs. We revisit the principal models for sensor node and network, make a special stress on the main features that identify WSNs, and describe some existing hardware platforms for sensor nodes. We then portray some examples of applications of WSNs, and finally present a review of the main optimization problems that are commonly acknowledged in the domain. Among these problems, we select two: the Wireless Sensor Network Layout problem (WSNL) and the Location Discovery problem (LD), and we add a third problem, the Radio Network Design (RND), which holds a resemblance to the node scheduling problem in WSNs. All of these problems are NP hard, and thus call for the use of metaheuristic techniques for their resolution.

1.3 Thesis contributions

In this section we briefly list the contributions of this thesis to the research field of WSNs or optimization. These contributions can be summed as follows:

- Thorough review of the State-of-the-art in WSNs, from a technological and application point of view.
- Establishment of a taxonomy of complex optimization problems found in WSN.
- Realistic modeling of representative problems selected: the WSN Layout optimization problem (WSNL) and the WSN Location Discovery problem (LD).
- Creation of a large sized real-world instance for the Radio Network Design problem (RND), and explanation of the problem's relation with the WSN domain.
- Proposal of CHC as a high-performing algorithm for the resolution of RND. Development of the multi-objective version of CHC, MOCHC, to tackle the multi-objective formulation of RND.
- Suggestion of an extension of the resolution procedure for RND that can be effectively applied to the scheduling problem in WSNs.
- Development of a new automatic migration tuning technique for distributed genetic algorithm that achieves results of similar quality as the best fixed migration schedule, while alleviating the cost of migration parameter tuning. Experimental evaluation on the RND problem.
- Development of a new problem-specific local improvement operator that improves the quality of the solutions produced by multi-objective optimization algorithms for the WSNL problem.
- Comparative study of the consistency of the main guiding functions in LD: the error norm and the likelihood functions.
- Proposal of a new two-stage solving procedure for LD that combines error norm and likelihood functions, and outperforms both individual functions.

1.4 Thesis organization

This thesis work is highly oriented towards the problem domain, and this reflects into its structure as a document. Thus, this thesis is divided into five parts, following this introduction. In the first part we present the fundamentals and basis for the work: the WSNs domain, Metaheuristics as a global family of resolution techniques, and the optimization algorithms that are selected to solve the problems. The second part is devoted to the first problem addressed, RND. Full reviews of the problem formulations, models, and existing literature are provided. Additionally, the relationship between this problem and the node scheduling problem is explained. The third part is devoted to the WSNL problem; all models, formulations and existing literature are reviewed. The fourth part corresponds to the LD problem; again the models, formulations and literature are reviewed. Finally, the fifth and last part of the thesis regroups the main conclusions drawn throughout the work and gives global comments about the work. We describe the contents of the chapters in greater detail below.

- **Part I: Fundamentals**

  Chapter 2 provides a general description of WSNs and sensor nodes. The main models at node and network levels are presented, and the main features that distinguish this kind of network from other ad hoc networks are listed, together with a short review of existing hardware platforms. We display

some notable examples of applications where these networks have been employed, and finish with a review of the main optimization problems that arise with this new field of research.

Chapter 3 gives an introduction to the research field of Metaheuristics, including the main concepts used, and the classifications that can be made over the techniques. We put special attention to some advanced mechanisms that are used in this work: multi-objective optimization and parallel metaheuristics, with a special review of a theoretical analysis of the convergence in distributed populations, that is later used as the basis for the development of our automatic migration tuning. Then, the selected algorithms that are employed to solve the problems are described in detail in Chapter 4.

- **Part II: Radio Network Design**

  Chapter 5 presents the Radio Network Design problem (RND). The principal models employed for the computation of the coverage are explained, and the existing literature is reviewed. Then, the node scheduling problem in sensor networks is presented and its relationship with the RND is explained. We propose an extension of the resolution process of RND that ca be applied to the node scheduling problem, and review the literature of the latter.

  In Chapter 6 we present our approach to solve the RND problem. We use two formulations for the problem objectives, a mono-objective and a multi-objective one, and two formulations for the problem type, a binary (parameterless) and an integer one (parameterized). We solve eight instances of different complexity. We introduce our automatic migration tuning technique, which is based on the theoretical analysis of the convergence process in distributed populations. Finally, we use the largest instance as the test bench to assess the effectiveness of the automatic migration tuning technique.

- **Part III: Wireless Sensor Network Layout**

  Chapter 7 presents the Wireless Sensor Network Layout problem (WSNL). We describe the different models existing for the coverage and the communications of the network, both at node level and network level. We introduce the concept of lifetime, and present its most common definitions. Finally, we provide a review of the existing literature for this problem.

  In Chapter 8 we describe our multi-objective formulation of the problem, with number of nodes and lifetime as (opposing) objectives, and coverage as a constraint. We describe the two types of genetic operator used: random and geographic. We propose the Proximity Avoidance Coverage-preserving Operator (PACO) as a local improvement operator that fixes local inefficiencies in the network design. Finally, we use three instances of different size to test the effectiveness and scalability of PACO.

- **Part IV: Location Discovery**

  Chapter 9 presents the Location Discovery problem (LD). We make a short review of the main ranging techniques that are used to generate the references used in LD, and describe the methods used to cope with existing errors in the measurements. Then, we present the most common functions that are used to guide the resolution process, namely the error norm and likelihood functions. Finally, we provide a review of the existing literature for this problem.

  In Chapter 10 we describe our formulation adopted for the LD problem, and the fitness function we take as the reference. We present the real data that serves as the basis to generate the problem instances and the models. We show the study performed for the two most popular guiding functions, and subsequently propose a novel two-stage approach to solve the LD problem. Finally, using 10 test instances (from as many sets of real data), we prove the effectiveness of our proposal through experimental evaluation.

- **Part V: Conclusions and future work**

  Chapter 11 contains a global review of the thesis work, and regroups the main conclusions drawn for the three problem instances. The thesis objectives and main contributions are discussed in view of the results obtained. Lastly, the future lines of research that can be pursued following the work presented here are briefly sketched and discussed.

Part I

FUNDAMENTALS OF THE THESIS

Chapter 2

Wireless Sensor Networks: Opportunities and Challenges

Observation is the key tool for experimental sciences. As Human knowledge and Technique have developed, so have the tools used to observe, measure, and sense. Recent advances in Micro Electro Mechanical Systems (MEMS) have brought the possibility to produce small devices with integrated computation and communication power. The engineering of increasingly smaller sized devices with multiple integrated capabilities have made possible for a new generation and a new conception of networked computation to arise. In this new conception, the central processing model that used to be found in classical computers (such as personal computers) no longer holds; instead, it is being replaced by a new paradigm where many small devices, with small/limited computing power, collaborate to obtain a composite computation power similar or superior to that of the centralized unit. When these small devices are given additional tools for interaction with their medium, a new concept appears: intelligent distributed sensing.

Wireless Sensor Networks (WSNs) are a relatively novel research field that constitutes a clear example of the aforementioned scenario. These networks are constituted by large numbers of power-constrained, performance-constrained devices known as sensor nodes. These sensors offer varying sensing capabilities, computing capabilities, and communicating capabilities. All in all, the nodes in a WSN are capable of little accomplishment when considered individually, yet they offer a wide spectrum of possibilities when acting collaboratively. The recent success of WSNs has led to the development of many hardware platforms, multiple sensing capabilities, and a plethora of application domains. Along with all of this came new design constraints and new operation objectives; in short, a set of novel and complex optimization problems.

This chapter presents and describes Wireless Sensor Networks (WSNs) as a novel monitoring tool. We describe the main characteristics and features that can be found (and that have to be looked after) in them, both at individual sensor node level, and at network level. The main types of sensing (devices) are presented, and a short review of existing hardware platforms (commercial models of sensor nodes) is provided. Afterwards, some notable examples of fields of application where WSNs have been successfully employed are presented. Lastly, we provide a short discussion on the main workhorses existing currently in WSNs, with special attention to the optimization problems that need to be solved in this new field.

2.1 Wireless Sensor Network description

Wireless Sensor Networks are a hot topic in research in many domains (electronics, communications, computer science, aerospace engineering, etc.). Many works have been published in the last years, and the numbers keep rising year after year. Domain-specific conferences, journals, and seminars have appeared. A number of surveys have defined the basis for current and future WSN: a survey focused towards new

possibilities and applications of WSNs can be found in [53], another survey with a more technical point of view, commenting software architectures and models, protocols and configurations of WSNs can be found in [4]. Other general surveys on WSNs architecture, protocol issues and potential applications can be found in [110]. We shall start by providing a short definition of a WSN:

Definition 1 (Wireless Sensor Network)**.** *A Wireless Sensor Network is an ad-hoc network of small autonomous collaborating devices with one or more sensing capabilities, known as* sensor nodes*, and that monitors a given phenomenon in a given place, known as the* sensing field*.*

In this section we will first describe the main component of WSNs, the sensor node. Then, we will present the architecture of a WSN.

2.1.1 The sensor node

The first thing to known, is that there is no *sensor node*, but many types of sensor nodes. Generally, sensor nodes are tailored for the application (purpose) or scenario they will be used in. Despite there being many common aspects shared by the majority of nodes, one can always find a given node for some specific application that does not share it. Nonetheless, this thesis is focused on general aspects of the WSNs, therefore we will describe the most common and widely acknowledged properties of the sensor nodes and networks.

Definition 2 (Sensor node)**.** *A* sensor node *is a small device with at least sensing, computation, and communication capabilities, that forms the basic component of a Wireless Sensor Network.*

In a little more detail, a sensor node is characterized by having the following set of attributes:

- It has a small size.
- It is inexpensive.
- It has one or more sensors[1].
- It has (limited) wireless communication capability.
- It has (limited) computation capability.
- It has (limited) storage capability.
- It has limited available energy.

This constitutes the basic overview of what a sensor node is. There are always exceptions, e.g., some sensors can have large energy supplies, others may be large, and in general the part about sensors being inexpensive is still far from becoming a reality. There are other optional capabilities a sensor node can have, some examples are energy harvesting (e.g., solar panels) to recharge their batteries, or mobility (they are called sensors and actuators).

Sensor nodes have an architecture that matches their properties. Figure 2.1 shows a diagram of the typical sensor node architecture. The basic blocks of a sensor node are the sensing unit (which contains the sensors), the processing unit (performs the computation), the transceiver (performs the wireless communications), and the power unit (stores the available energy). Additionally, the diagram in Figure 2.1 includes three optional modules that add extra capabilities to the node: one for energy harvesting (the "Power generator"), mobility (the "Mobilizer"), and location discovery (the "Location finding system").

[1] *Sensors* and *sensor nodes* are often confounded concepts. A sensor is an electronic device that is capable of measuring some physical magnitude (e.g, temperature) and produce an electronic output accordingly. A sensor node is a larger device that acts as a node of a WSN and is a platform containing, among other things, one or more sensors.

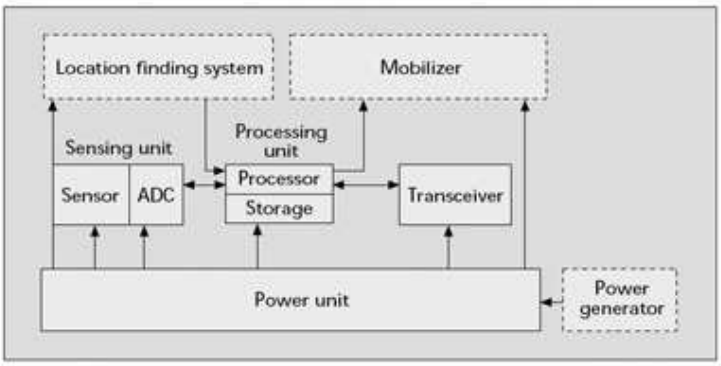


Figure 2.1: Basic block diagram of a sensor node.

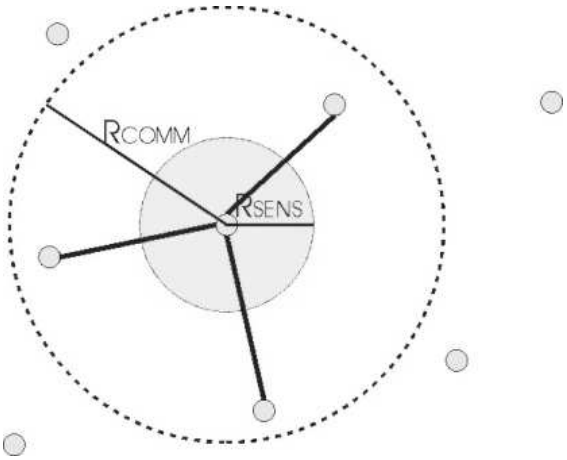


Figure 2.2: Basic model of a sensor node with communication radius R_{COMM} and sensing radius R_{SENS}.

When used in an optimization problem, the sensor node generally needs to be modeled. The most basic sensor node model is a device located at a given point (x, y) with sensing capacity over a disk of radius R_{SENS} (the *sensing radius* or *sensing range*), and communication capacity in a disk of radius R_{COMM} (the *communication radius* or *communication range*). Figure 2.2 illustrates the basic model for a sensor node that will be considered in this work. The node at the center covers the shaded area, and communicates through a direct link with the nodes connected to it by a boldface line. Both R_{COMM} adn R_{SENS} are shown. This model is further explored and developed in Chapter 7.

2.1.2 Wireless Sensor Network architecture

A WSN is thus a collection of sensor nodes that form a connected structure. Since the purpose of a WSN is almost always to monitor a given terrain, the sensed information must be made accessible to the network user or administrator. For this, there is generally an access point to the network, a special node known as *sink* or High Energy Communication Node (HECN). This node acts as a gateway to the network, and is a special node; generally, its energy resources are considered limitless (from the WSN operation perspective). All the nodes in the network must be able to communicate (via direct link or multi-hop paths) to the HECN, any node that does not have a path to the HECN is considered *disconnected* from the network and is not taken into account for sensing purposes.

There are two basic architectures found in WSNs: the flat or plain architecture, in which all nodes are strictly similar, and the hierarchical structure where nodes have different roles depending on which tier they are in. We briefly describe both next.

- **Flat**. In a WSN with a *flat* or *plain* architecture, all sensor nodes are equal in hierarchy. Figure 2.3a shows a simple example of the hierarchy of a plain WSN with 12 nodes. Ad-hoc WSN are plain networks, since all nodes are equal. Note that the networks are still multihop, hence the communication topology of the network does not match the hierarchical organization.

- **Hierarchical** ([92, 108]). In a hierarchical or *clustered* WSN, there are a subset of the sensor nodes that have higher hierarchy than the rest, they are known as cluster-heads. The basic clustered network is two-tiered, but the concept can be generalized to an arbitrary number of tiers, with cluster-heads of different levels. Figure 2.3b shows an example of a two-tiered clustered WSN. In this case the communication structure often matches the hierarchical structure of the network, however this is not always the case, as in some protocols the cluster-heads form a multihop network to connect themselves to the HECN. Cluster heads can be special nodes with higher capacity than regular nodes (specially in terms of energy and computation power), or they can be regular nodes, elected with some protocol; in the latter case, the role of cluster head is normally rotated among the nodes in the network so that the extra load is shared[2].

In this work, we focus our attention on flat networks.

2.1.3 Special characteristics of Wireless Sensor Networks

WSNs are specifically designed to offer a new set of monitoring capabilities, yet they have to deal with the inherent constraints imposed by the environment, the budget, or the sensor node's technical limitations. All of this accumulates to generate a wide field of optimization. Some of the main issues that need to be dealt with when using WSNs come from the operation requirements of the WSN, and others come from the hardware constraints of the sensor nodes.

A WSN is conceived to work in an automated way, that is, without the requirement of a human operator; this feature is often referred to as *unmanned operation*. Thus, sensor nodes have to perform a set of tasks ranging from topology discovery and routing, to periodic sensing and data transmission, and all of this in a completely unattended manner.

WSNs are expected to contain large numbers of nodes in a short future, ranging from hundreds to even thousands. Furthermore, many WSNs are deployed in hostile environments. Therefore, it is impractical and sometimes even impossible to physically access the nodes, hence any unpredicted situation must be handled by the network and the individual nodes on their own. Besides, the hostility of the environment may cause frequent node failure (or even destruction); the WSN must then be able to recover from one (or more) node failure, that is, it must have fault tolerance. In some cases it is also possible that more nodes

[2]It can be argued that networks where clusterhead roles rotate among the nodes are in fact a hybrid between flat and hierarchical, sin all nodes will have the same behavior by the end.

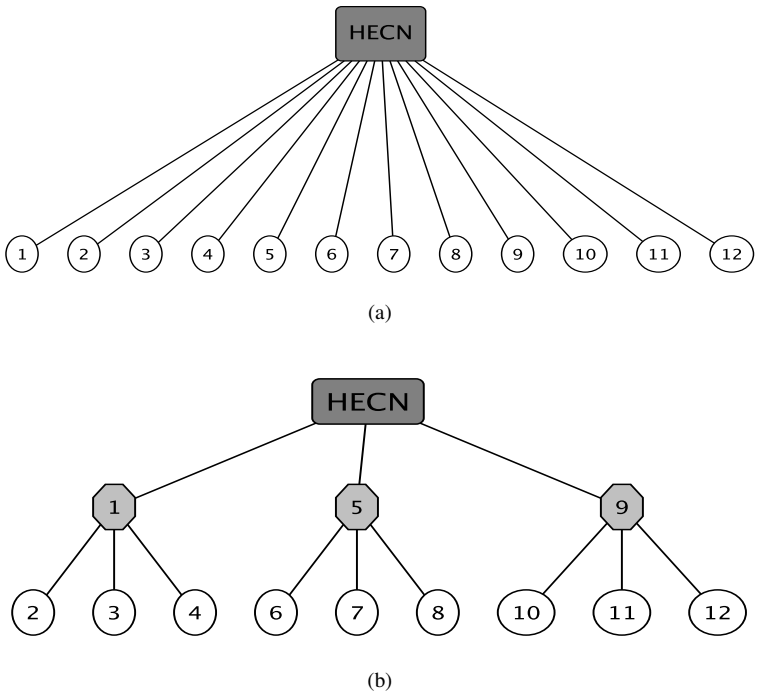


Figure 2.3: Network hierarchical architectures of a WSN: (a) plain and (b) clustered.

will be deployed *after* the WSN has been active for a period of time (in order to replace lost nodes, or to expand the monitoring capabilities of the network); in these cases the network must be prepared to detect on-the-fly and incorporate newly arrived nodes into itself, that is, it must reconfigure itself (and operate) dynamically.

A WSN can be required to provide a timely picture of the monitored phenomenon. Therefore, there must be a maximum latency between an event happens, and the event is reported. This simple requirement supposes a complex resolution that affects the sensing policy, the data processing, and the communication of information through the network.

Some of the main special characteristics found in WSNs are listed below:

- They have an *unmanned* operation, i.e., they don't require the presence/assistance of a human operator.
- They can host very large numbers of nodes; scalability of the operation thus becomes a major issue.
- They need to have automatic configuration mechanisms.
- They have to respond to dynamic changes in the environment or the network (apparition/disappearance of nodes).
- They need lightweight general operation mechanisms, preferably locally distributed (for routing, processing, etc.).
- They are *data-centric*, i.e., they provide an abstraction from the nodes structure/topology.
- They provide time and spatial integration of the sensed data (it should be possible to construct a time-dependent map representation of the sensed data).

- They are often deployed in harsh environments, or places with difficult access. Hence, they cannot be maintained.
- They need to operate for long periods of time unmaintained.
- They may not have the possibility of energy restoration.
- They have to be unobtrusive, and should operate undetected.
- They need security mechanisms (authentication, encryption, etc.).

2.2 Sensor types

There are many physical magnitudes that one would like to monitor, and a WSN may just be the ideal system. For it, many different types of sensor exist that can be integrated into the sensor node platform. We dedicate this section to provide a review of the main types of sensors and the physical magnitudes they sense. We provide a brief list of the main categories of sensing ([137]):

- **Mechanical sensors**: that rely on physical contact. Among these are:
  - Piezoresistive effect-based sensors. They convert an applied strain to a change in resistance that can be sensed using electronic circuits. The relationship is $\Delta R/R = S\epsilon$, where R is the resistance, ϵ the strain and S the gauge factor.
  - Piezoelectric effect-based sensors. They convert an applied stress (force) to a charge separation or potential difference; this effect is reversible. The change in voltage V is given by $\Delta V = k\Delta F$, where ΔF is the change in force and k is the sensitivity.
  - Tunneling sensing. The sensing depends on the exponential relationship between the tunneling current I and the tip/surface separation z given by $I = I_0 \exp(-kz)$, where k is a sensitivity factor.
  - Capacitive sensors. They typically have one fixed plate and one movable plate. When a force is applied to the movable plate, the change in capacitance C is given as $\Delta C = \epsilon A/\delta d$, with d the resulting displacement, A the area, and ϵ the dielectric constant.
- **Magnetic and electromagnetic sensors**: they react to magnetic or electromagnetic fields (do not require contact). Among these are:
  - Magnetoresistive sensors. The Hall voltage induced in a plate of thickness T is given by $V_H = RI_xB_z/T$, with R the Hall coefficient, I_x the current flow in direction x, and B_z the magnetic flux density in the z direction.
  - Magnetic field sensors. They can be used to detect the remote presence of metallic objects.
- **Thermal sensors**: they sense the temperature. Among these are:
  - Thermo-mechanical transduction sensors. They react to changes in temperature T, by exhibiting (linear) thermal expansion of the form $\Delta L/L = \alpha\Delta T$, with L the length and α the coefficient of linear expansion.
  - Thermoresistive sensors. The resistance R changes with the temperature T following the relation approximately given by $\Delta R/R = \alpha_R\Delta T$, with α_R the temperature coefficient of resistance.

  - Thermocouples. If a circuit consists of two different materials joined together at each end, with one junction hotter than the other, a current flows in the circuit. This generates a Seebeck voltage given approximately by $V \approx \alpha(T_1 - T_2) + \gamma(T_1^2 - T_2^2)$, with T_1, T_2 the temperatures at the two junctions.
  - Resonant temperature sensors. They rely on the fact that single-crystal SiO2 exhibits a change in resonant frequency depending on temperature change.

- **Optical transducers**: they convert light to a measurable quantity. Among these are:
  - Photoelectric effect sensors. The *photoelectric effect* causes one electron to be emitted at the negative end of a pair of charged plates for each light photon of sufficient energy. This causes a current to flow.
  - Photoconductive sensors. Photons generate carriers that lower the resistance of the material.

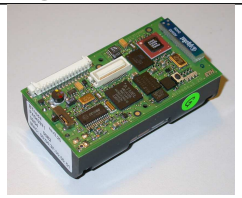
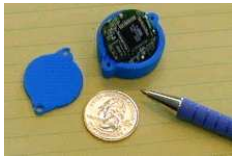
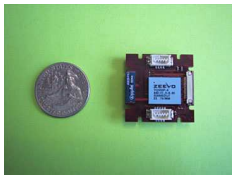
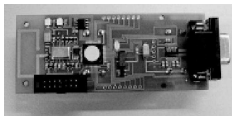
- **Chemical and biological transducers**: these cover a very wide range of devices that interact with solids, liquids, and gases of all types. They have been effectively used for pollution detection. Among these are:
  - Chemiresistor sensors. They have two interdigitated finger electrodes coated with specialized chemical coatings that change their resistance when exposed to certain chemical challenge agents.
  - Metal-Oxide Gas sensors. They rely on the fact that adsorption of gases onto certain semiconductors greatly changes their resistivities
  - Electrochemical Transducers. They rely on currents induced by oxidation or reduction of a chemical species at an electrode surface. These are among the simplest and most useful of chemical sensors.
  - Biosensors. These devices have a biochemically active thin film deposited on a platform device that converts induced property changes (e.g., mass, resistance) into detectable electric or optical signals.

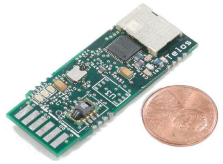
- **Acoustic sensors**: include those that use sound as a sensing medium. Doppler techniques allow the measurement of velocities. Ultrasound can be used to get information about mechanical machinery vibrations. Among these are:
  - Acoustic Wave sensor. The surface of the device can be coated with a chemically or biologically active thin film. On presentation of the measurand to be sensed, adsorption might cause the mass m to change, resulting in a frequency shift given by the Sauerbrey equation $\Delta f = k f_0^2 \Delta m / A$, with f_0 o the membrane resonant frequency, constant k depending on the device, and A the membrane area.

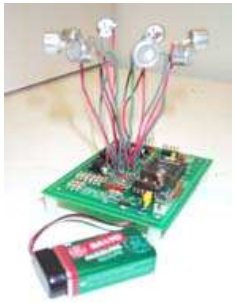
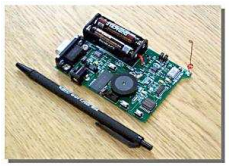
2.3 Commercial sensor nodes

There are many publicly available lists and reviews of the different commercial models of sensor nodes ([22, 88]). The large number of different existing platforms (and increasing by the day) makes impossible to give a full review of the choices commercially available to design a WSN. Therefore, in Table 2.1 we present a short review of the principal node models, with their main features (regarding their computation, memory, communications, and sensing).

Table 2.1: Commercially available sensor nodes.

Node	Image	Microcontroller	Memory	Radio	I/O Sensors
BTnode		Atmel ATmega 128L (8 MHz @ 8 MIPS)	64+180 Kbyte SRAM, 128 Kbyte Flash ROM, 4 Kbyte EEPROM	Chipcon CC1000 operating in ISM Band (433-915 MHz)	UART, SPI, I2C, GPIO, ADC, Clock, Timer, LEDs Standard Molex 1.25mm Wire-to-Board and Hirose DF17 Board-to-Board connectors
Dot		ATMEL Microcontroller 916 MHz	1KB RAM 8-16KB Flash		
SunSPOT		ARM 920T	512K RAM + 4MB Flash	802.15.4	Accelerometer, Temperature and Light sensors, 8 LEDs
Imote		ARM core 12MHZ	64KB SRAM, 512 KB Flash	Bluetooth with the range of 30 m	USB, UART connector
Rene		ATMEL8535	512B RAM 8K Flash	916 Mhz radio with bandwidth of 10 Kbps	
Eyes		MSP 430F149 (5 MHz @ 16 Bit)	8 Mbit	RFM TR1001 hybrid radio transceiver	UART, AD and I/O, JTAG interface and sensor board with compass, accelerometer, temperature sensor, light sensor, pressure sensor, microphone nad push button lines

Node	Image	Microcontroller	Memory	Radio	I/O Sensors
Mica2		ATMEGA 128L	4K RAM 128K Flash	315, 433 or 868/916Mhz Multi-Channel transceiver with 38 Kbaud	
MicaZ		ATMEGA 128	4K RAM 128K Flash	802.15.4/ZigBee compliant RF transceiver	
Telos		Motorola HCS08	4K RAM	250Kbps	USB and Ethernet
Tmote Sky		Texas Instruments MSP430	10k RAM and 48k Flash	250kbps 2.4GHz IEEE 802.15.4	Integrated Humidity, Temperature, and Light sensors
PicoNode		Strong ARM 1100	4Mb DRAM, 4mB fLASH	Bluetooth, Proxim	2 sensor boards: Board 1: temperature, humidity, light, and sound. Board 2:temperature, acceleration, magnetic fields and provisions for GPS
Stargate		Intel PXA255	64KNSRM	802.11	@PCMICA/CF, com ports, USB, Ethernet

Node	Image	Microcontroller	Memory	Radio	I/O Sensors
weC		Atmel AVR AT90S2313		RFM TR1000 RF	
iBadge		Atmel ATMEGA and TI DSPC5416		Bluetooth radio (64kBPS)	Temperature, pressure, humidity, magnometer,accelerometer. Acoustic in/out+DSP
Medusa		40MHz ARM THUMB	1MB Flash, 136KB RAM	RFM (the same as MICA)	ultrasound transceivers
UAMPS		StrongARM SA-1100	16Mb RAM, 512KB ROM	SA-1100 ISM 2.45 GHZ with 1Mbps and range up to 15 meters	Seismic and acoustic sensor
SpotON		MC68EZ328 “Dragonball”		RFMonolithics TR1000 916Mhz	Location sensing platform containing accelerometer and infrared detector

2.4 Applications of Wireless Sensor Networks

WSNs have been intensively used in many and varied applications. The applications that have been found to be best suited for WSNs are those involving a difficult access, hostile environment, automated operation, unobtrusive operation, requiring close distance measurements covering wide regions, etc. Some interesting surveys on WSNs and their application scenarios can be found in [71, 126]. Some links to possible applications for WSNs are provided in [110]. We present in this section a brief review of the main application fields where WSNs have been used.

- **Military applications.** Military applications have been the main driving force behind the development of WSNs, especially at its earliest stages. This kind of application includes reconnaissance, enemy detection and tracking. For instance, a counter-sniper detection system for urban warfare scenarios was presented in [135], and a WSN for target data acquisition in [169]. Multi-vehicle tracking in the context of a pursuit-evasion game using a WSN is described in [193].

- **Surveillance.** Surveillance applications are generally target detection and tracking; they are mostly related to military or high security areas, where all access and presence have to be controlled, but is slowly being generalized to less critical scenarios, such as personnel localization inside a facility, or object inventorying in large depots. A target detection application with classification and tracking is presented in [14].

- **Civil engineering.** This kind of application includes mostly the monitoring of the health state of large structures, such as skyscrapers or bridges. In [220], a WSN is proposed for structural health measurements. Bridge monitoring in railways is performed by the Brimon WSN in [37].

- **Health applications.** Health applications involve the use of small sensors to measure biometric quantities in human beings. The sensors have very strict safety requirements, and can be external or internal (placed inside the body). Some examples of WSNs in this field are health data monitoring in [227], or biomedical sensors for artificial retina in [183].

- **Human-centric applications.** Also known as ambient intelligence. This kind of application can be described as "comfort" applications, and indoor environmental. They are not critical applications, but can be offered as services to customers. Such applications include domotics (automatic air conditioning control), object location in a house/office as in [225], and the like. Another possibility is traffic control, as in [94] where a WSN is deployed in a highway to monitor the traffic. A WSN for monitoring a heating and air conditioning plant is presented in [203].

- **Industry.** Industrial WSNs may be used to control the state of heavy machinery, automatic quality control of products, or physical measurements to monitor some chemical/mechanical process (humidity, heat, light, pressure, pH, etc.) ([23]). For example, the condition of pumps at gas stations, oil and gas drilling monitoring ([100]), the heat of the rolls used in paper production, and the vibrations in semiconductor fabrication ([125]) have been all monitored by WSNs.

- **Agriculture.** WSNs have also been employed in agriculture; they generally sense the humidity and richness of the soil and provide automated management of crops (for lower costs). They have been used to monitor vineyards in [27]. Another example is [130], where a WSN is used for precise agriculture (and the main problems encountered are listed and described as lessons learned).

- **Environmental.** Environmental applications mainly refer to the unobtrusive observation of (possibly endangered [19]) animal or vegetal species, their environmental conditions and the variations of that environment (possibly by the hand of man). Habitat monitoring by WSNs is discussed in [150]; in it, a WSN that was deployed in Great Duck Island (Maine) for habitat monitoring of the storm petrel is described. Another environmental application of WSNs is forest fire prevention ([162]). A WSN

to monitor the environmental conditions (light, humidity) in the forests of California, home to the redwood trees, is described in [53].

- **Disaster relief.** Disaster relief are on-the-fly deployed WSNs that can be used to locate missing persons or detect safe and dangerous areas in an earthquake, flooding, volcano eruption, etc. Other WSNs may be used as prevention devices, in order to predict such events before they happen to allow the proper measures to be taken. For instance, in [212] a WSN is deployed on an active volcano. A WSN called FLOODNET is presented in [116] to measure water level at the River Crouch, in order to predict possible flooding.

- **Exploration.** WSNs can also be deployed on remote places, or environments of difficult access. For instance, the ocean floor, or outer space ([67]).

2.5 Optimization problems in Wireless Sensor Networks

Optimization problems have naturally arisen within the context of WSNs, mainly due to the confrontation between the application purposes, and the design constraints, many of which are novel. There already exists a number of surveys about the optimization problems defined (and solved) for WSNs. These problems have been defined and approached in many heterogeneous ways; some authors have proposed specific heuristics to solve given problems, while others employ linear programming, and finally metaheuristics have come into the scene. An interesting survey on evolutionary approaches applied to WSNs is [166]; in it, five optimization problems are identified as the most representative of WSNS: resource management for lifetime optimization, position estimation of nodes, multi-sensor fusion, energy-efficient routing, and node placement and layout optimization. Location discovery and routing are also presented and discussed in [18]. We note that in this thesis work we address two of the main problems found in WSNs directly, namely the position estimation of nodes (location discovery, chapters 9 and 10) and the node placement and layout optimization (WSN layout, chapters 7 and 8); additionally we solve a problem that is closely related to the resource management (radio network design, chapters 5 and 6, which can be expanded to solve the scheduling problem).

We propose the following general classification of optimization problems for WSNs that can be found in the literature:

- **Sensor deployment and layout optimization**. This problem amounts to designing the geographic configuration of the WSN; that is, the locations where the nodes will be placed. Some WSNs can be placed manually, therefore the position of each node should be carefully calculated; in other networks the nodes cannot be placed individually, but there is some deployment mechanism that can be coarsely designed according to their resulting node positions (e.g., dropping nodes from a plane, one can decide when and where to drop more nodes, when and where to drop less). The number of nodes may be a fixed or a variable amount. The aimed objectives in this problem include the coverage achieved (to be maximized), the energy efficiency, a balanced network topology, and the number of deployed sensor nodes (to be minimized). Proper literature review for this problem is shown in Section 7.5.

- **Location discovery or localization in WSNs**. Location information is a basic feature in sensor networks. It holds an enormous importance for routing, scheduling, and of course spatial integration of the sensed data. For instance, in an intrusion detection mechanism, one needs to know *where* the intruder has been detected; similarly, in a forest fire prevention application, the *location* of the fire is an essential information in order to coordinate the response. Typically, a small subset of the nodes know their location (either by manual insertion, GPS, or other methods); these nodes are called *beacon* or *anchor* nodes. The GPS is not generalized to all the nodes in the network due to its

high cost. The rest of nodes use then a set of references consisting in node-to-node and node-to-beacon measured distances, from which they derive their own location. The sought objectives are to minimize the node location errors and to have topology consistency (that is, the node's relative locations are consistent with the network topology). Proper literature review for this problem is shown in Section 9.6.

- **WSN scheduling for lifetime optimization**. The scheduling or resource allocation in a WSN is the process that assigns tasks to processing elements, or more generally to available task-performing resources. Since WSNs are generally considered to provide a monitoring of the field that is continuous from a timeline point of view, the tasks are considered to be the sensing process at the different time periods, and the resources are the nodes themselves. Therefore, scheduling corresponds to assigning time periods during which a node will be sensing and communicating (it is said to be in an *active* state), while the rest of the time, it will be in a power-saving mode (which is usually called *sleeping* state). From a network point of view, the scheduling consists in selecting at each time the subset of nodes that need to be active, while the rest can be put to sleep. A scheduling may be performed in a centralized manner (using all the information of the network, producing a single schedule and broadcasting it to all nodes), or in a distributed manner (each node, or a local subgroup of nodes, independently decide their own schedule using only local information). The objectives of the scheduling process are to maximize the network's lifetime by having the nodes sleep for the maximum possible time, subject to the constraints of maintaining the coverage and connectivity. Proper literature review for this problem is shown in Section 5.4.2. A problem related to scheduling, the optimal sampling rate assignment, is discussed in [116, 144].

- **Inter-sensor synchronization in WSNs**. This problem consists in obtaining a shared synchronization among the nodes. There are methods aimed at obtaining a shared clock reference among the nodes. Alternatively, other procedures are aimed at providing clock-independent functioning, that is, processes that work properly even in the absence of such a synchronization. The objective of synchronization is to minimize the difference among the different nodes internal clocks, or minimize the effect of that difference. Global clock synchronization for WSNs is discussed in [200]. Fault-tolerant clock synchronization is discussed in [200], and energy-efficient synchronization in [201]. Synchronization-robust angular location discovery is discussed in [167].

- **Topology control**. This problem includes several aspects. In topology control the network must make sure it stays connected; one available mechanism it has for this is transmission power control. Another aspect of topology control is that the resulting connectivity graph of the WSN has to be energy-efficient and adequate for the routing protocol that shall be used. For instance, if geographic forwarding is used (see Section 7.3.2), a topology with many geographic local optima is very harmful. Clustering methods are a special kind of topology control mechanisms. The SPAN protocol was proposed for topology control in [38]. In [140], a pruning method is devised for WSN in order to reduce the number of links to alleviate the charge due to neighbors in some nodes and the bandwidth stress; Relative Neighborhood Graph, Gabriel Graph, Delaunay triangulation, and Yao Graph are used. A planar spanning of the network is searched in [26]. Fault-tolerant topology control is discussed in [89, 109, 154]. The effect of radio irregularity and the resulting asymmetric links is studied in [233]. Asymmetric link detection is also performed in [91]. Transmission power control is employed in [96, 234].

- **Routing in WSNs**. The routing problem amounts to deciding the routing strategy for data collecting that is employed within the WSN (we recall that the sensed data have to be accessible to the user or the application), and for the coordination among the nodes. Routing algorithms generally include both a general behavior specification *and* a distributed (node-sized) implementation to achieve that behavior. This problem has been extensively addressed in the literature, and is one of the five problems

described in [166]. The main concerns in routing are the energy-efficiency (to maximize the lifetime), the latency of the data transmissions (to obtain timely responses), and the reliability (so that the data is effectively transmitted even though the medium is intrinsically unreliable). Some proposed routing protocols are Geographic Forwarding (GF, [112]), Symmetric Geographic Forwarding (SGF, [233]), Ad-hoc On-demand Distance Vector routing (AODV, [176]), Dynamic Source Routing (DSR, [99]), Efficient Greedy Geographic Routing ([228]), Greedy Perimeter Stateless Routing (GPSR, [26, 39]). A routing algorithm to balance energy consumption and latency is proposed in [128]. Geographic-informed routing is discussed in [222]. Other specific routing algorithms can be found in [199]. Routing in heterogenous WSNs is discussed in [149]. Other approaches to data collection in WSNs include the use of a mobile observer that retrieves directly (by direct link communication) the information from the nodes in [36].

- **Data-fusion in WSNs**. This problem is often intertwined with the routing; data-fusion calls for the establishment of a technique that allows one or a local set of nodes to encode their data, typically to reduce the raw bandwidth and storage requirements, and also to provide them with some preprocessing that alleviates posterior post-processing. Data-fusion ranges from simple data compression at node level to complex multi-sensor data integration, where data of multiple natures are combined into higher-level indicators. Data-fusion is one of the five main problems in WSNs ([166]). The main objective sought by data fusion is to reduce the amount of transmitted data through the network without losing information, in order to reduce the required energy for transmissions and to respect the bandwidth restrictions of the WSN. A review of the most common aggregation methods in WSNs is found in [45]. Two data coding schemes, joint-entropy and Slepian-Wolf, are used in [80]. In [91], data-fusion is used to reduce false alarms. Correlated information is considered in [90, 199]

- **Security in WSNs**. Security issues can be crucial in WSNs depending on the application at hand. For instance, in a military application, the WSN must be absolutely protected from attacks, unauthorized intrusions or hearings. Therefore, the network must provide mechanisms for authentication of the nodes and users, data encryption and intrusion detection[3]. The issue of security in Location Discovery is discussed in [41, 132]. In the first, three types of attack are contemplated (Sybil, wormhole, and compromised entity); in the second, signal strength attacks are described. Security from radio attacks (by producing interference) is granted by a *surfing* strategy in [221].

We note that there is a list of issues that are transversal to most of these problems, that is, many problems are concerned with them and have to cope with similar constraints. Many issues are related and have a common ground, rendering this separation not entirely clear at times. The main issues of this kind are the following:

- **Energy efficiency and lifetime**. This is perhaps the most widespread concern in WSN. Due to size and cost constraints, sensor nodes have small batteries (generally, one or two AA batteries) and thus small energy; nevertheless, WSNs need to provide continuous service for long periods, without access to extra energy resources (energy harvesting is scarcely used –and generally provides little power–, and batteries are almost never replaced in a WSN). Therefore, energy conservation during the WSN operation is one of the most important objectives in practically all applications of WSNs, for lifetime maximization. There is an almost endless list of works where the lifetime of the network is one of the optimization objectives, to name a few [36, 52, 80, 95, 96, 128, 140, 144, 184, 188, 199, 234]. Lifetime from a theoretical point of view is studied in [62, 103, 201, 229]. A survey on energy-efficient coverage problems is made in [34]. Energy-harvesting nodes are used in [97]. A general

[3] Intrusion detection can be understood in two ways in the context of WSNs. The first one refers to the physical detection, location and tracking of a target in the secure area corresponding to the sensor field, much like the alarm of a house detects a robber; hence this is a design objective of some WSNs and has to be maximized. The second sense corresponds to the infiltration of a virtual agent in the logical domain of the WSN, getting access to restricted information or information modification, command capacity, or a combination of the previous; hence, this is an undesired risk that WSNs are exposed to and has to be minimized.

survey on energy sources for sensor nodes is given in [211]. Energy models are reviewed or proposed in [24, 46, 114, 129, 155, 189].

- **Detection probability and false alarm rate**. The detection probability is a coverage estimator and is usually one of the parameters considered in WSN problems (for instance, in scheduling or layout optimization) for maximization. However, sometimes, due to the random nature and the noise of measurements, an undesired effect know as false positives or false alarm can happen: in this case the network notifies an event when in reality there is nothing happening; this has to be minimized. False positives (false alarm) are considered in [144, 191]. Detection rates are considered in [96, 134, 135, 231].

- **Latency of the WSN response**. Many WSNs are deployed to monitor critical events and need real-time response to events. This means that the time between an event happens and that event is reported by the WSN must be upper bounded (the value of the bound should depend on the nature of the monitored phenomenon). The latency of the WSN has the following components: latency of the detection (includes scheduling and sensor hardware-intrinsic latency), latency of the process (includes distributed detection and data fusion), and latency of the reporting (includes data transmission, routing). A broadcast protocol for controlled latency is proposed in [184]. Latency is also optimized in [128]. Other works that consider latency are [62, 135, 221, 225].

- **Distributed operation**. The distributed execution of the algorithms (employing only local resources) is a fundamental item for the scalability of WSNs; considering that these networks are expected to contain in the order of hundreds or thousands of nodes, scalability is not a minor concern. Some problems require intrinsically distributed solutions, like routing or data aggregation; others, like layout optimization, scheduling or location discovery admit both centralized and distributed approaches. There are also several degrees of distribution, for instance an application can be clustered (each clusterhead performs a "centralized" process for its cluster), or it can be entirely distributed (every node performs its own computation). Distributed solutions are provided for location discovery in [49, 141, 174, 208], for scheduling in [95], for sampling rate in [116]. Cooperative solutions for location discovery are commented in [138], for detection in [107]. Clustered solutions for congestion control are shown in [108].

- **Efficient use of reduced computation and storage capabilities**. The proposed solutions, save exceptions, will have to run on the execution platform provided by a sensor node (since they *should be* distributed whenever they can, according to the previous point). Therefore, software tools and solutions (including operating systems, protocols and algorithms) should be as lightweight as possible.

- **Efficient use of reduced communication capabilities (transmission range, bandwidth)**. On the one hand, WSNs operate in narrow bands, on the other hand, sensor nodes have limited storage capacity and need to transmit their data frequently; these two facts, combined with the large number of nodes and their continuous monitoring operation mode, can easily cause congestion in the network: too much data is transmitted at the same time, and the transmission medium does not have enough capacity for it. This problem affects mainly the sampling and routing protocols (they cannot store large routing tables). Controlled information rates for congestion control, and limited bandwidth are considered in [36, 42, 43, 108, 140, 144, 230].

- **Robustness against harsh conditions and hostile environments**. As said before, WSNs are often deployed in unfriendly or hostile environments (sometimes they are deployed *precisely* because the environment is hostile and other methods cannot be used). This produces, among other undesired effects, unreliable communications. Radio irregularity is studied in [233]. Lossy links are considered in [228]. Robustness in data transmissions (reliability) is considered in [184], protection against radio interference is proposed in [221], signal strength attacks to location discovery are commented in [41].

- **Robustness against node errors and node failures**. This issue is closely related to the preceding one, since the harsh conditions of the environment can cause node malfunction, failure or even destruction. Furthermore, sensor nodes are cheap and need to be manufactured in large numbers, hence the (physical) robustness is not their strong point: sensor nodes are considered to be error-prone devices (they may break, they may run out of energy and stop functioning). As said before, a common assumption in WSNs is that, for one reason or another, nodes cannot be replaced. Therefore, the WSN must be able to cope with the failure of nodes in the best possible way. Localization errors are considered in [219]. Robustness against node failures in ensured by k-connectivity in [96, 134]. Robustness in clock synchronization is discussed in [200]. A cut detection method is proposed in [191], to detect cuts in the WSN. Robustness in clustering and topology control is considered in [109, 154].

We defer a more thorough discussion about the use of metaheuristics and other optimization techniques to solve the optimization problems found in WSNs selected to be solved in this thesis, to their corresponding chapters.

2.6 Conclusions

In this chapter, we have presented a global picture of WSNs. We have first defined the sensor network and sensor node concepts, and described the most currently used models from the architectural and functional points of view for both entities; a special stress is put in highlighting the specific features found in WSNs that make them different from other kinds of ad hoc network, including both their possibilities, and their constraints. We have provided a review of existing sensor types, and of available sensor nodes. Then, we have briefly discussed the main types of applications for which WSNs are being used to the date, providing some examples for each category. Finally, we have presented and described the principal optimization problems that have to be solved in order to achieve an operational WSN.

Chapter 3

Metaheuristics

A heuristic technique or heuristic (from the greek "$Eu\rho i\sigma\kappa\omega$" meaning "find" or "discover") is a *rule of thumb*, an educated guess, a simple and intuitive technique that produces close to optimal solutions for a given complex problem. Sometimes that problem exact solution is unknown, or maybe the technique for obtaining it is just too heavy (i.e., time consuming); in these cases, having a heuristic comes in handy, since it offers a solution to the problem, albeit not necessarily the optimal one.

However practical, heuristics have the drawback of being problem-specific techniques, thus a good heuristic for some given problem will help little when solving a different problem. Therefore, a necessity for more general-purpose optimization techniques arises that made way for the appearance of higher-level, general-purposed techniques which capitalized on most of the heuristics benefits: Metaheuristics. This new brand of techniques is fairly recent, with the initial developments in the field being during the late 50's and 60's, and has taken root until becoming a wide research tool, and even a research topic. Metaheuristics are generally conceived as high level heuristics, and use some heuristics at atomic step level, which in turn belong to a bigger, more complex process.

This chapter serves as a general introduction to metaheuristics. In it, the metaheuristic techniques utilized to solve the optimization problems of this work will be presented for their latter description in the next chapter. The application of metaheuristics to multi-objective problems, as well as the issue of distributed metaheuristics are explained. Finally, the quality indicators and the statistical tests used to assess their significance are described.

3.1 Definition of a metaheuristic

A metaheuristic is a high level technique or algorithm for solving complex optimization problems. They are stochastic algorithms which do not guarantee to obtain the optimal solution of the problem, but when properly tuned obtain near-optimal solutions with bounded computation effort. We shall begin with a formal definition of optimization. Assuming, without loss of generality, a *minimization* case, the definition of an *optimization problem* is as follows:

Definition 3 (Optimization problem)**.** *An optimization problem is defined as a pair (S, f), where $S \neq \emptyset$ is called the solution space (or search space), and f is a function named* objective function *or* fitness function*, defined as:*

$$f : S \to \mathbb{R} \ . \tag{3.1}$$

Thus, solving an optimization problem consists in finding a solution $i^* \in S$ such that:

$$f(i^*) \leq f(i), \quad \forall \ i \in S \ . \tag{3.2}$$

Note that assuming either maximization or minimization does not restrict the generality of the results, since an equivalence can be made between the two cases in the following manner ([15, 86]):

$$\max\{f(i)|i \in S\} \equiv \min\{-f(i)|i \in S\} \ . \tag{3.3}$$

Depending on the domain where S belongs, we can speak of *binary optimization problems* ($S \subseteq \mathbb{B}^*$), *integer* ($S \subseteq \mathbb{N}^*$), *continuous* ($S \subseteq \mathbb{R}^*$), or *heterogeneous* ($S \subseteq (\mathbb{B} \cup \mathbb{N} \cup \mathbb{R})^*$).

A simple classification of the optimization methods used throughout the history of computer science is shown in Figure 3.1. Initially, the techniques can be classified into exact and approximate. The exact techniques, which are based on the mathematical extraction of the optimal solution, or an exhaustive search until the optimum is found, guarantee the optimality of the solution obtained. These techniques present some drawbacks, however. The time they require, though bounded, is generally very long, especially for NP-hard problems. Furthermore, it is not always possible to find such an exact technique for every problem. This makes exact techniques not to be the right choice in many occasions, since both their time and memory requirements can become unreasonably high for large problems. Therefore, approximate techniques have been often used by the international research community in the last few decades. These methods sacrifice the guarantee of finding the optimum in favor of providing some satisfactory solution within reasonable time.

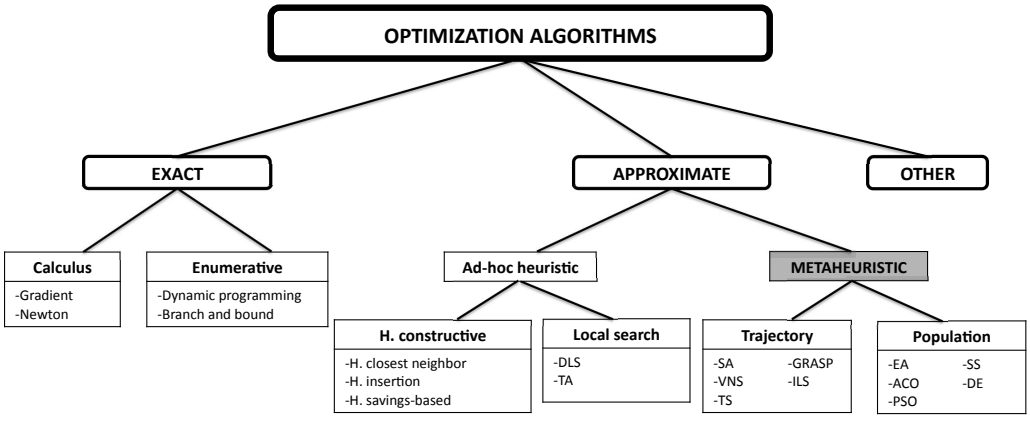


Figure 3.1: General classification of the optimization techniques.

Among approximate algorithms, one can find two types: *ad hoc* heuristics, and metaheuristics. We focus this chapter on the latter. *Ad hoc* heuristics can in turn be divided between *constructive heuristics* and *local search methods*.

Constructive heuristics are usually the swiftest methods. They construct a solution from scratch by iteratively incorporating components until a complete solution is obtained, which is returned as the algorithm output. Finding some constructive heuristic can be easy in many cases, but the obtained solutions are of low quality. In fact, designing one such method that actually produces high quality solutions is a nontrivial task, since it mainly depends on the problem, and requires thorough understanding of it. For example, in problems with many constraints it could happen that many partial solutions do not lead to any feasible solution.

Local search or gradient descent methods start from a fully complete solution. They rely on the concept of *neighborhood* to explore a part of the search space defined for the current solution until they find a *local optimum*. The neighborhood of a given solution s, denoted as $N(s)$ is the set of solutions (neighbors) that can be reached from s through the use of a specific modification operator (generally referred to as *movement*). A local optimum is a solution equal or better than any other solution in its own neighborhood. The process of exploring the neighborhood, finding and keeping the best neighbor, is repeated in a process until the local optimum is found. Complete exploration of a neighborhood is often unapproachable, therefore some modification of the generic scheme has to be adopted. Depending on the movement operator, the neighborhood varies and so does the manner of exploring the search space, simplifying or complicating the search process as a result.

Lastly, during the 70's, a new class of approximate algorithms appeared whose basic idea was to combine several heuristic methods at a higher level to achieve an efficient and effective search of the search space. These techniques are called *metaheuristics*. The term was first introduced by Glover ([83]). Until the term was ultimately adopted by the scientific community, these techniques were named *modern heuristics* ([178]). This class of algorithm includes many diverse techniques such as ant colony, evolutionary algorithms, iterated local search, simulated annealing or tabu search. A survey of metaheuristics can be found in [21, 85]. Out of the different descriptions of metaheuristics that can be found in the literature, some fundamental properties can be highlighted:

- Metaheuristics are general strategies or templates that guide the search process.
- Their goal is to provide an efficient exploration of the search space to find (near) optimal solutions.
- Metaheuristics are not exact algorithms and their behavior is generally non deterministic (stochastic).
- They may incorporate mechanisms to avoid visiting non promising regions of the search space.
- Their basic scheme has a predefined structure.
- Metaheuristics may use specific problem knowledge for the problem at hand, by using some specific heuristic controlled by the high level strategy.

In short, a metaheuristic is a high level strategy that employs different methods to explore the search space. In other words, a metaheuristic is a general template for a non deterministic process that has to be filled with specific data from the problem to be solved (solution representation, specific operators to manipulate them, etc.), and that can tackle problems with high dimensional search spaces.

In these techniques, the success depends on the correct balance between *diversity* and *intensity*. The term diversity refers to the evaluation of solutions in distant regions of the search space (with some distance previously defined for the solution space); it is also known as *exploration* of the search space. The term intensity refers to the evaluation of solutions in small bounded regions, or within a neighborhood (*exploitation* of the search space). The balance between these two opposed aspects is of the utmost importance, since on the one hand the algorithm has to find quickly the most promising regions (exploration), and on the other hand those promising regions have to be thoroughly searched (exploitation).

We can distinguish two kinds of search strategy in metaheuristics. First, there are "intelligent" extensions of local search methods (trajectory-based metaheuristics in Figure 3.1). These techniques add some

mechanism to escape local optima to the basic local search method (which would otherwise stick to it). Tabu Search (TS), Iterated Local Search (ILS), Variable Neighborhood Search (VNS) or Simulated Annealing (SA) are some techniques of this kind. These metaheuristics operate with a single solution at a time, and one (or more) neighborhood structures. A different strategy is followed in Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), or Evolutionary Algorithms (EAs). These techniques operate with a set of solutions at any time (called population, swarm or colony depending on the case), and use a learning factor as they, implicit or explicitly, try to grasp the correlation between design variables in order to identify the regions of the search space with high-quality solutions (population-based techniques in Figure 3.1). In this sense, these methods perform a biased sampling of the search space.

Formally, a metaheuristic is defined as a tuple of elements that, depending on how they are defined, result in some technique. This formal definition was developed in [148] and later extended in [44].

Definition 4 (Metaheuristic)**.** *A metaheuristic $\mathcal{M}$ is a tuple consisting of eight components as follows:*

$$\mathcal{M} = \langle \mathcal{T}, \Xi, \mu, \lambda, \Phi, \sigma, \mathcal{U}, \tau \rangle \ , \tag{3.4}$$

where:

- *$\mathcal{T}$ is the set of elements operated by the metaheuristic. This set contains the search space, and in many cases they both coincide.*
- *$\Xi = \{(\xi_1, D_1), (\xi_2, D_2), \ldots, (\xi_v, D_v)\}$ is a collection of v pairs. Each pair is formed by a state variable of the metaheuristic and the domain of said variable.*
- *μ is the number of solutions operated by $\mathcal{M}$ in a single step.*
- *λ is the number of new solutions generated in every iteration of $\mathcal{M}$.*
- *$\Phi : \mathcal{T}^\mu \times \prod_{i=1}^{v} D_i \times \mathcal{T}^\lambda \to [0,1]$ represents the operator that produces new solutions from the existing ones. The function must verify for all $x \in \mathcal{T}^\mu$ and for all $t \in \prod_{i=1}^{v} D_i$,*

$$\sum_{y \in \mathcal{T}^\lambda} \Phi(x,t,y) = 1 \ . \tag{3.5}$$

- *$\sigma : \mathcal{T}^\mu \times \mathcal{T}^\lambda \times \prod_{i=1}^{v} D_i \times \mathcal{T}^\mu \to [0,1]$ is a function that selects the solutions that will be manipulated in the next iteration of $\mathcal{M}$. This function must verify for all $x \in \mathcal{T}^\mu$, $z \in \mathcal{T}^\lambda$ and $t \in \prod_{i=1}^{v} D_i$,*

$$\sum_{y \in \mathcal{T}^\mu} \sigma(x,z,t,y) = 1 \ , \tag{3.6}$$

$$\begin{aligned} &\forall y \in \mathcal{T}^\mu, \sigma(x,z,t,y) = 0 \vee \\ &\quad \vee \sigma(x,z,t,y) > 0 \wedge \\ &(\forall i \in \{1,\ldots,\mu\} \bullet (\exists j \in \{1,\ldots,\mu\}, y_i = x_j) \vee (\exists j \in \{1,\ldots,\lambda\}, y_i = z_j)) \ . \end{aligned} \tag{3.7}$$

- *$\mathcal{U} : \mathcal{T}^\mu \times \mathcal{T}^\lambda \times \prod_{i=1}^{v} D_i \times \prod_{i=1}^{v} D_i \to [0,1]$ represents the updating process for the state variables of the metaheuristic. This function must verify for all $x \in \mathcal{T}^\mu$, $z \in \mathcal{T}^\lambda$ and $t \in \prod_{i=1}^{v} D_i$,*

$$\sum_{u \in \prod_{i=1}^{v} D_i} \mathcal{U}(x,z,t,u) = 1 \ . \tag{3.8}$$

- $\tau : \mathcal{T}^{\mu} \times \prod_{i=1}^{v} D_i \rightarrow \{false, true\}$ *is a function that decides the termination of the algorithm.*

The previous definition recollects the typical stochastic behavior of metaheuristics. In fact, the functions Φ, σ and $\mathcal{U}$ should be considered as conditional probabilities. For instance, the value of $\Phi(x, t, y)$ is the probability to generate the offspring vector $y \in \mathcal{T}^{\lambda}$, since the current set of individuals in the metaheuristic is $x \in \mathcal{T}^{\mu}$, and its internal state is given by the state variables $t \in \prod_{i=1}^{v} D_i$. One can notice that the constraints imposed over the functions Φ, σ and $\mathcal{U}$ enable them to be considered as functions that return the conditional probabilities.

Definition 5 (State of a metaheuristic)**.** *Let $\mathcal{M} = \langle \mathcal{T}, \Xi, \mu, \lambda, \Phi, \sigma, \mathcal{U}, \tau \rangle$ be a metaheuristic and $\Theta = \{\theta_1, \theta_2, \ldots, \theta_{\mu}\}$ the set of variables containing the solutions handled by the metaheuristic. We shall note as $first(\Xi)$ the set of state variables of the metaheuristic, $\{\xi_1, \xi_2, \ldots, \xi_v\}$. A* state *$s$ of the metaheuristic is a pair of functions $s = (s_1, s_2)$ with:*

$$s_1 : \Theta \rightarrow \mathcal{T}, \tag{3.9}$$

$$s_2 : first(\Xi) \rightarrow \bigcup_{i=1}^{v} D_i \ , \tag{3.10}$$

where s_2 verifies

$$s_2(\xi_i) \in D_i, \ \ \forall \xi_i \in first(\Xi) \ . \tag{3.11}$$

We denote with $\mathcal{S}_{\mathcal{M}}$ the set of all states of a metaheuristic $\mathcal{M}$.

Finally, once the state of a metaheuristic is defined, we can define its dynamics.

Definition 6 (Dynamics of a metaheuristic)**.** *Let $\mathcal{M} = \langle \mathcal{T}, \Xi, \mu, \lambda, \Phi, \sigma, \mathcal{U}, \tau \rangle$ be a metaheuristic and $\Theta = \{\theta_1, \theta_2, \ldots, \theta_{\mu}\}$ the set of variables containing the solutions handled by the metaheuristic. We denote as $\overline{\Theta}$ the tuple $\langle \theta_1, \theta_2, \ldots, \theta_{\mu} \rangle$ and as $\overline{\Xi}$ the tuple $\langle \xi_1, \xi_2, \ldots, \xi_v \rangle$. We extend the definition of a state in order to apply it onto element tuples, that is, we define $\overline{s} = (\overline{s}_1, \overline{s}_2)$ where:*

$$\overline{s}_1 : \Theta^n \rightarrow \mathcal{T}^n \ , \tag{3.12}$$

$$\overline{s}_2 : first(\Xi)^n \rightarrow \left(\bigcup_{i=1}^{v} D_i \right)^n \ , \tag{3.13}$$

and

$$\overline{s}_1(\theta_{i_1}, \theta_{i_2}, \ldots, \theta_{i_n}) = \langle s_1(\theta_{i_1}), s_1(\theta_{i_2}), \ldots, s_1(\theta_{i_n}) \rangle \ , \tag{3.14}$$

$$\overline{s}_2(\xi_{j_1}, \xi_{j_2}, \ldots, \xi_{j_n}) = \langle s_2(\xi_{j_1}), s_2(\xi_{j_2}), \ldots, s_2(\xi_{j_n}) \rangle \ , \tag{3.15}$$

for $n \geq 2$. We call r a successor state of s if there exists $t \in \mathcal{T}^{\lambda}$ such that $\Phi(\overline{s}_1(\overline{\Theta}), \overline{s}_2(\overline{\Xi}), t) > 0$,

$$\sigma(\overline{s}_1(\overline{\Theta}), t, \overline{s}_2(\overline{\Xi}), \overline{r}_1(\overline{\Theta})) > 0 \ \ and \tag{3.16}$$

$$\mathcal{U}(\overline{s}_1(\overline{\Theta}), t, \overline{s}_2(\overline{\Xi}), \overline{r}_2(\overline{\Xi})) > 0 \ . \tag{3.17}$$

We denote with $\mathcal{F}_{\mathcal{M}}$ the binary relationship "being successor of" defined for the set of states of a metaheuristic $\mathcal{M}$. That is, $\mathcal{F}_{\mathcal{M}} \subseteq \mathcal{S}_{\mathcal{M}} \times \mathcal{S}_{\mathcal{M}}$, and $\mathcal{F}_{\mathcal{M}}(s, r)$ if r is a successor state of s.

Definition 7 (Execution of a metaheuristic)**.** *An execution of a metaheuristic $\mathcal{M}$ is a finite or infinite sequence of states $s_0, s_1, \ldots$ in which $\mathcal{F}_{\mathcal{M}}(s_i, s_{i+1})$ for all $i \geq 0$ and:*

- *if the sequence is infinite, it verifies that $\tau(s_i(\overline{\Theta}), s_i(\overline{\Xi})) = false$ for all $i \geq 0$ and*
- *if the sequence is finite, it verifies that $\tau(s_k(\overline{\Theta}), s_k(\overline{\Xi})) = true$ for the last state s_k and, $\tau(s_i(\overline{\Theta}), s_i(\overline{\Xi})) = false$ for all $i \geq 0$ such that $i < k$.*

3.2 Classification of metaheuristics

There are many ways to classify metaheuristics ([21]). Depending on the chosen features one can obtain different taxonomies: nature inspired vs. non nature inspired, memory based vs. memoryless, one or several neighborhood structures, etc. One of the most popular classifications distinguishes *trajectory based* metaheuristics from *population based* ones. Those of the first type handle a single element of the search space at a time, while those of the latter work on a set of elements (the population). This taxonomy is graphically represented in Figure 3.2, where the most representative techniques are also included. The next two sections describe these kinds of metaheuristic in turn.

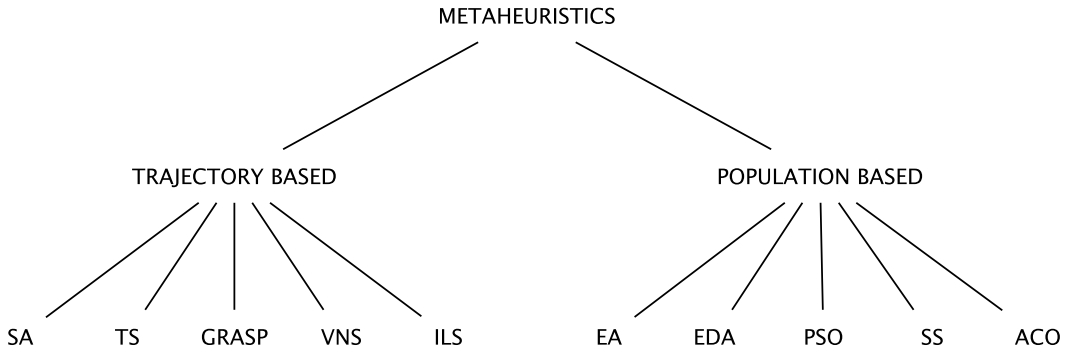


Figure 3.2: Classification of metaheuristics.

3.2.1 Trajectory based metaheuristics

This section serves as a brief introduction to trajectory based metaheuristics. The defining feature of these methods is the fact that they start from a single solution, and, by successive neighborhood explorations, update that solution, describing a trajectory through the search space. According to the notation in the Definition 4, this corresponds to $\mu = 1$. Most of the algorithms of this kind are extensions of simple local search methods, which receive some additional mechanism for escaping local optima. This results in a more complex stopping condition than the simple detection of a local optimum. Some widely used stopping criteria are completing some predefined number of iterations, finding some acceptable solution, or reaching some stagnation point.

Simulated Annealing (SA)

Simulated Annealing (SA) is one of the oldest techniques among metaheuristics and the first algorithm with an explicit strategy for escaping local optima. Its origins can be found in a statistical mechanism, called *metropolis* ([157]). The main idea in SA is to simulate the annealing process of a metal or crystal. SA was first introduced in [117]. To avoid getting stuck in a local optimum, the algorithm always allows the selection of a solution with worse *fitness* value than the current one with some probability. The mechanism works as follows: in each iteration a solution s' is extracted from the neighborhood $N(s)$ of current solution s; if s' has better fitness value than s, then s is discarded and s' is kept instead, otherwise s is replaced by s' only with a given probability that depends on a dynamic parameter T called temperature, and the difference between the fitness values of the two solutions, $f(s') - f(s)$.

Tabu Search (TS)

Tabu Search (TS) is one of the metaheuristics that has been most successfully used to solve combinatorial optimization problems. The basics of this method were introduced in [83], and they rely on the ideas formulated in [82], where the technique and its components are properly explained. The main idea in TS is the use of an explicit search history (short term memory), that serves both for escaping local optima and for enhancing the diversity of the search process. This short term memory is called the tabu list, and keeps record of the last visited solutions, preventing the algorithm from visiting them again. At the end of each iteration, the best solution among the allowed ones is included in the list. From the perspective of the implementation, keeping a list of full solutions is inefficient due to wasted memory consumption. Therefore, a commonly adopted alternative is to register the *movements* performed by the algorithm instead. In any case, the elements in the list can be used to filter the neighborhood, producing a reduced set of eligible solutions named $N_a(s)$. Storing movements instead of complete solutions is more efficient, but causes a loss of information as well. In order to avoid this problem, an aspiration criterion is defined that permits the inclusion of a solution in $N_a(s)$ despite that solution being in the tabu list. The most widely used aspiration criterion is to permit solutions with better fitness values than the best fitness found so far.

GRASP

The *Greedy Randomized Adaptive Search Procedure* (GRASP, [74]) is a simple metaheuristic that combines constructive heuristics with local search. GRASP is an iterative procedure with two phases: first, a solution is constructed, second, the solution undergoes an improvement process. The improved solution is the final result of the search process. A randomized heuristic is used for the construction of the solution in the first phase. Step by step, different components c are added to the partial solution s^p, initially empty. Each added component is randomly selected from a restricted list of candidates (RCL). This list is a subset of $N(s^p)$, the set of permitted components for the partial solution s^p. The components of the solution in $N(s^p)$ are sorted according to some problem dependent function η in order to generate the list. The RCL list consists of the α best components in the set. In the extreme case of $\alpha = 1$, only the best component found is added to the list, thus resulting in a greedy construction method. In the other extreme, $\alpha = |N(s^p)|$, the component is chosen in a totally random way among all available components. Hence, α is a key parameter that determines how the search space is going to be sampled. The second phase of the algorithm consists in a local search method to improve the previously generated solutions. A simple local search method can be employed, or some more complex technique like SA or TS.

Variable Neighborhood Search (VNS)

The *Variable Neighborhood Search* (VNS) is a metaheuristic proposed in [161], that uses an explicit strategy to switch among different neighborhood structures during the search. It is a very generic algorithm with many degrees of freedom to design variations or particular instances. The first step is to define the set of neighborhood descriptions. There are many ways this can be done: from random selection up to complex mathematical equations deduced using problem knowledge. Each iteration contains three phases: selection of a candidate, improvement phase, and finally, the movement. During the first phase, a neighbor s' is randomly chosen in the k^{th} neighborhood of s. This solution s' acts then as the starting point for the second phase. Once the improvement process is over, the resulting solution s'' is compared with the original, s. If s'' is better then it becomes the current solution and the neighborhood counter is reset ($k \leftarrow 1$); if it is not better, then the process is repeated for the next neighborhood structure ($k \leftarrow k + 1$). The local search can be considered as the intensity factor, whereas the switches among neighborhoods can be considered as the diversity factor.

Iterated Local Search (ILS)

The *Iterated Local Search* (ILS) metaheuristic ([145, 198]) is based on a simple yet effective concept. In each iteration, the current solution is perturbed and, to this new solution, a local search method is applied, to improve it. An acceptance test is applied to the local optimum obtained from the local search to determine whether it will be accepted or not. The perturbation method has an obvious importance: if it is not disruptive enough, the algorithm may still be unable to escape the local optimum; on the other side, if it is too disruptive, it can act as a random restarting mechanism. Therefore, the perturbation method should generate a new solution that serves as the starting point for the local search, but not so far away from the current solution as to be a random solution. The acceptance criterion acts as a balance method, since it filters which new solutions can be accepted depending on the search history and the characteristics of the local optimum.

3.2.2 Population based metaheuristics

Population based methods are characterized by working with a set of solutions at a time, usually named population, unlike trajectory based methods, which handle a single solution. Population based methods have generally $\mu > 1$ and/or $\lambda > 1$.

Evolutionary Algorithms (EAs)

Evolutionary Algorithms (EAs) are loosely inspired on the theory of the natural evolution of the species. The techniques in this wide family follow an iterative stochastic process that operates a population of solutions, each solution being referred to within this context as *individual*. Initially, the population is generated in a random way (or with some constructive heuristic). The general template of an EA has three phases, named after their natural equivalents: selection, reproduction and replacement. The whole process is repeated until some stopping criterion is met (generally, after a certain number of operations has been performed). The selection phase selects the fittest individuals from the current population, to be recombined later during the reproduction phase. The resulting individuals from the recombination are modified by a mutation operator. Finally, the new population is formed with individuals from the current one, and/or the best newly generated individuals (according to their fitness values). This new population is used as the current population in the next iteration of the algorithm. A well known example of EA is the Genetic Algorithm (GA).

Estimation of Distribution Algorithms (EDAs)

The *Estimation of Distribution Algorithms* (EDAs, [165]) have similar behaviors to the previously presented EAs, and many authors even consider EDAs as a special kind of EA. Like EAs, EDAs operate on a population of candidate solutions, but, unlike them, do not use recombination and mutation to generate the new solutions, but a probability distribution mechanism instead. Graphic probabilistic models are commonly used tools to represent in an efficient manner the probability distributions when working with EDAs. Some authors ([131, 175, 196]) propose the use of bayesian networks to represent the probability distributions in discrete domains, while Gaussian networks are most often applied for continuous domains ([214]).

Scatter Search (SS)

The *Scatter Search* (SS, [84]) is another metaheuristic whose basic principles were presented in [82], and is currently receiving an increasing deal of attention from the research community ([127]). The algorithm's fundamental idea is to keep a relatively small set of candidate solutions (called the reference set, or *RefSet* for short), characterized by hosting diverse (distant in the search space) high-quality solutions. Five components are required for the complete definition of SS: initial population creation method, reference set

generation method, subsets of solutions generation method, solution combination method, and improvement method.

Ant Colony Optimization (ACO)

The *Ant Colony Optimization* (ACO, [63, 64]) algorithms are inspired by the foraging behavior of real ants in the search for food. This behavior can be described as follows: initially, ants explore the surrounding area of their nest or colony in a random fashion. As soon as an ant finds a food source, it starts carrying that food to the nest; as it does this, the ant continuously deposits a chemical substance known as pheromone in its path. This substance can be detected by other ants, thus guiding them to the food. This indirect communication among ants also serves to find the shortest path between the nest and the food. ACO methods intend to simulate this behavior to solve optimization problems. These techniques have two main phases: construction of a solution following a single ant's behavior, and update of the artificial pheromone trace. There is no *a priori* planning or synchronization between the phases, which can even be done simultaneously.

Particle Swarm Optimization (PSO)

The *Particle Swarm Optimization* (PSO, [115]) algorithms are inspired in the social behavior of bird flocks or fish schools. PSO keeps a set (called *swarm*) of solutions (called *particles*), initialized randomly throughout the search space. Each particle has position and speed, both constantly changing during the search. The movement of a particle is determined by its current speed, and the relative position of the particle itself and some reference particles in its neighborhood. Within PSO, the *neighborhood of a article* is defined as a subset of particles from the swarm; this concept of neighborhood is different from the one previously used in trajectory based methods. The neighborhood of a particle can be *global* when all particles are considered neighbors, or *local* when only close particles are considered neighbors.

3.3 Metaheuristics for multi-objective problems

Most of the real world optimization problems require to optimize two or more objective functions which usually are in conflict with each other. Problems of this kind are usually referred to as *multi-objective Optimization Problems* (MOPs). Due to the lack of accurate methodological approaches, MOPs have been tackled as mono-objective optimization problems (e.g., making use of aggregative functions) in the past. However, the working principles guiding mono-objective and multi-objective optimization are completely different. When solving MOPs, we are interested in the best possible trade-offs (or compromises) among the different objectives (i.e., solutions in which it is not possible to improve one objective without worsening another). This is so because, in the absence of any further information, all the objectives of a MOP are considered equally important. Thus, the solution to a MOP is not a single solution but a set of them. As a consequence, to solve a MOP typically involves two different phases: on the one hand, to optimize the objective functions, and, on the other hand, a decision making procedure to choose the most accurate solution (giving a set of preferences or external context, [47]). Paying attention on how both phases are faced, multi-objective optimization techniques can be classified as follows ([48]):

- *A priori*: when decisions are taken before finding the solutions.
- *Progressive*: when both, the decision making and the search, are integrated.
- *A posteriori*: the decision making takes place after finding the solutions.

Each group has different advantages and drawbacks which make each different technique more accurate than the others in some particular scenarios, and vice-versa ([47, 57]). Nevertheless, the first two groups are strongly influenced by the decision of an expert (*decision maker*) who determines the degree of importance

of an objective over the others, which could restrict the search space in a arbitrary way failing to find the optimal solution to the problem. On the other hand, *a posteriori* techniques make a wider exploration of the search space in order to compute as many compromise solutions as possible. Once this phase has finished, the decision making procedure takes place. This last group of algorithms has been intensively used in the field of metaheuristics and, particularly, in the field of evolutionary computation ([47, 57]). Specifically, the most advanced *a posteriori* techniques make use of the *Pareto optimality* concept ([173]), which as a matter of fact is the approach adopted for the MOPs considered in this work. Hence, this section is structured as follows. First, we introduce some basic concepts for multi-objective optimization, from the perspective of Pareto optimality. The next subsection presents the goals that all multi-objective optimization techniques should achieve. Finally, the third subsection is aimed at discussing different design aspects which should be considered when designing metaheuristics following those principles.

3.3.1 Basic concepts

In this section, we include some background on multi-objective optimization. Informally, a MOP can be defined as the problem consisting in finding a vector of decision variables which satisfies a set of constraints and optimizes a number of objective functions. Those functions define a set of performance criteria which are in conflict with each other. Thus, the term "optimization" refers to the search of such a vector, which has acceptable values for all the objective functions ([172]).

From the mathematical point of view, the formulation of a MOP extends the classic definition of mono-objective optimization by considering the existence of two or more objective functions. Thus, there is not a single solution but a set of them. This set is found by considering the Pareto Optimality Theory ([68]). More formally, a general multi-objective optimization problem (MOP) can be defined as follows:

Definition 8 (MOP)**.** *A multi-objective optimization problem is defined as a tuple $\langle S, f, g, h\rangle$, where $S \neq \emptyset$ is called the solution space (or search space), $f = [f_1, f_2, \ldots, f_k]$ is a vectorial function, where $f_i : S \to \mathbb{R}$, are the objective functions, and $g = [g_1, g_2, \ldots, g_m]$ and $h = [h_1, h_2, \ldots, h_p]$ are vectorial functions, where $g_i : S \to \mathbb{R}$ and $h_i : S \to \mathbb{R}$ are the constraint functions. Thus, solving an optimization problem consists in finding a set of solutions $X^* \subseteq S$ such that, for all $x^* \in X^*$:*

$$f_j(x^*) \leq f_j(x), \quad \forall\ x \in S\ . \tag{3.18}$$

for some $1 \leq j \leq k$, subject to:

$$g_i(x^*) \leq 0 \quad i = 1, 2, \ldots, m\ , \tag{3.19}$$

$$h_i(x^*) = 0 \quad i = 1, 2, \ldots, p\ , \tag{3.20}$$

where $g_i, h_j : S \to \mathbb{R}$, $i = 1, ..., m$, $j = 1, ..., p$ are the constraint functions of the problem.

Definition 9 (Pareto dominance)**.** *Given two vectors $\vec{x}, \vec{y} \in \mathbb{R}^k$, we say that $\vec{x} \leq \vec{y}$ if $x_i \leq y_i$ for $i = 1, ..., k$, and that $\vec{x}$* **dominates** *$\vec{y}$ (denoted by $\vec{x} \prec \vec{y}$) if $\vec{x} \leq \vec{y}$ and $\vec{x} \neq \vec{y}$.*

Definition 10 (Non-dominance)**.** *We say that a solution $x \in S$ is* **non-dominated** *with respect to S, if there does not exist another $x' \in S$ such that $f(x') \prec f(x)$.*

Figure 3.3 illustrates graphically both concepts, Pareto dominance and Non-dominance. Specifically, it shows two distinct sets of solutions computed for a multi-objective problem where the two objective functions, f_1 and f_2, are to be minimized. Since both objectives are equally important, it is not trivial to decide which solution is better. Considering the previous definitions, we can say that a is better than b in the picture on the left as $f_1(a) < f_1(b)$ and $f_2(a) < f_2(b)$, i.e., a is better in all the objective functions; thus, we say that a *dominates* b ($a \prec b$). The same can be said with respect to a and c: $f_1(a) < f_1(c)$ and $f_2(a) < f_2(c)$, thus $a \prec c$. Let's compare now solutions b and c. In this case, we can observe that c is better

than b in the f_1 objective function, but b is better than c in f_2 ($f_2(b) < f_2(b)$). According to Definition 9, we cannot say that b dominates c, nor c dominates b. In this case, the solutions are said to be non-dominated within respect to one another. In the right side graphic of Fig. 3.3, we show four non-dominated solutions, where none can be said to be better than the others.

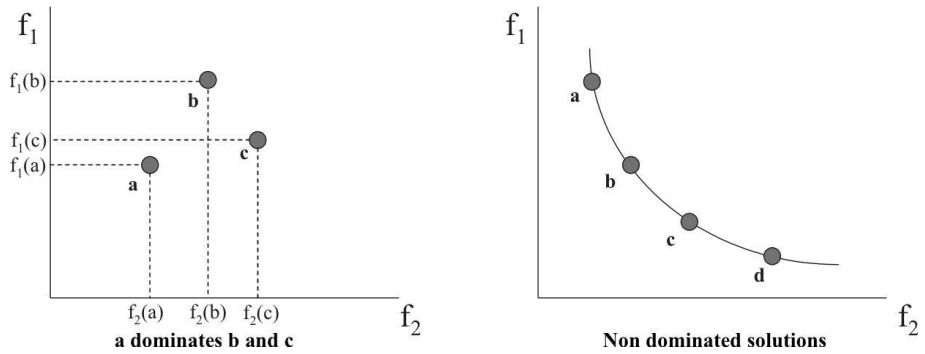


Figure 3.3: Dominance in multi-objective optimization: (left) solution 'a' dominates 'b' and 'c', (right) non dominated solutions.

Thus, solving a MOP consists in computing the set of solutions that dominates every other point in the solution space; this means that the solutions in that set are optimal for that problem. Formally:

Definition 11 (Pareto Optimality)**.** *We say that a solution $x^* \in \mathcal{F}$ ($\mathcal{F} \subseteq S$ is the feasible region) is* **Pareto optimal** *if it is non-dominated with respect to $\mathcal{F}$.*

Definition 12 (Pareto Optimal Set)**.** *The* **Pareto Optimal Set** *$\mathcal{P}^*$ is defined by:*

$$\mathcal{P}^* = \{x \in \mathcal{F} | x \text{ is Pareto-optimal}\} \ . \tag{3.21}$$

It is important to note that while Pareto optimal solutions belong in the variable space (S), their vector components belong in the objective space ($\mathbb{R}^k$). Those solutions are usually referred to as *non inferior*, *acceptable*, or *efficient*. The Pareto front can then be defined, as:

Definition 13 (Pareto Front)**.** *The* **Pareto Front** *$\mathcal{PF}^*$ is defined by:*

$$\mathcal{PF}^* = \{f(x) \in \mathbb{R}^k | x \in \mathcal{P}^*\} \ . \tag{3.22}$$

That is to say, the Pareto front is composed of the values in the objective space corresponding to the solutions of the Pareto optimal set. Generally, it is not easy to find an analytic expression of the curve or surface containing those points, and in many cases it is downright impossible. Figures 3.4 and 3.5 show the formulation and the corresponding Pareto fronts of problems Binh2 and DTLZ4 ([47]). In the first case, it is a bi-objective problem having two decision variables x_1 and x_2, and two constraints, $g1$ and $g2$. As for the DTLZ4 problem, it has three objective functions and no restriction.

3.3.2 Goals when solving MOPs

When solving a MOP the main goal is to compute its Pareto optimal set (and Pareto front). In theory, this set (front) could contain a large number of (or even infinitely many) points. In practice, a usable approximate solution will only contain a limited number of points; thus, an important goal is that the corresponding front should be as close as possible to the exact Pareto front and uniformly spread, otherwise, it would not be

$$
\begin{array}{lcll}
\text{Min } F & = & (f_1(\vec{x}), f_2(\vec{x})) & \\
f_1(\vec{x}) & = & 4x_1^2 + 4x_2^2 & \\
f_2(\vec{x}) & = & (x_1 - 5)^2 + (x_2 - 5)^2 & \\
\text{Subject to:} & & & \\
g1(\vec{x}) & = & (x_1 - 5)^2 + x_2^2 - 25 & \leq 0 \\
g2(\vec{x}) & = & -(x_1 - 8)^2 - (x_2 + 3)^2 + 7.7 & \leq 0 \\
0 & \leq & x_1 & \leq 5 \\
0 & \leq & x_2 & \leq 3
\end{array}
$$

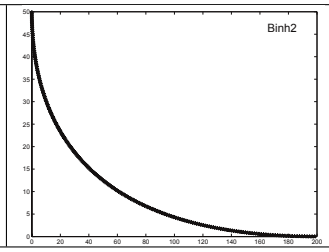


Figure 3.4: Formulation and Pareto front for the Bihn2 problem.

$$
\begin{array}{lcl}
f_1(x) & = & (1 + g(x_M)) \cos(x_1^{\alpha} \frac{\pi}{2}) \cos(x_2^{\alpha} \frac{\pi}{2}) \\
f_2(x) & = & (1 + g(x_M)) \cos(x_1^{\alpha} \frac{\pi}{2}) \sin(x_2^{\alpha} \frac{\pi}{2}) \\
f_3(x) & = & (1 + g(x_M)) \sin(x_1^{\alpha} \frac{\pi}{2}) \\
 & & 0 \leq x_i \leq 1, \quad i = 1, 2, \ldots, n \\
g(x_M) & = & \sum_{x_i \in x_M} (x_i - 0.5)^2 \\
n & = & 12 \\
x_M & = & x_3, ..., x_{12} \\
\alpha & = & 100
\end{array}
$$

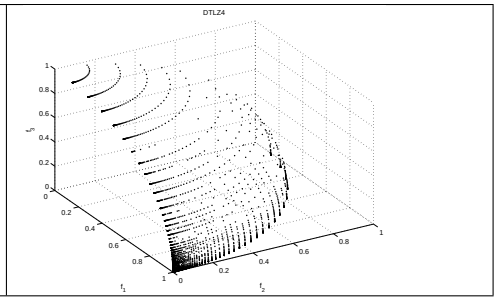


Figure 3.5: Formulation and Pareto front for the DTLZ4 problem.

very useful to the decision maker. Closeness to the Pareto front ensures dealing with optimal solutions, while a uniform spread of the solutions means a good exploration of the search space, and that no regions are left unexplored. Thus, we seek Pareto fronts meeting the following goals:

1. As close as possible to the optimal Pareto front (convergence).

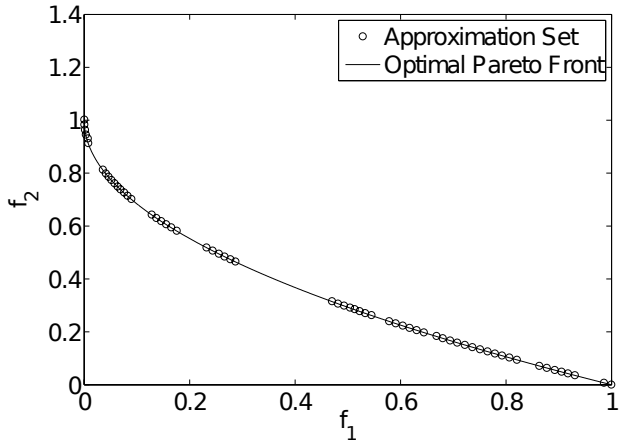
2. As diverse as possible (diversity).

While the first goal, convergence towards the optimal solution, is a requirement in every optimization problem (independently of the number of objectives), the second one is specific of problems involving the optimization of more than one objective.

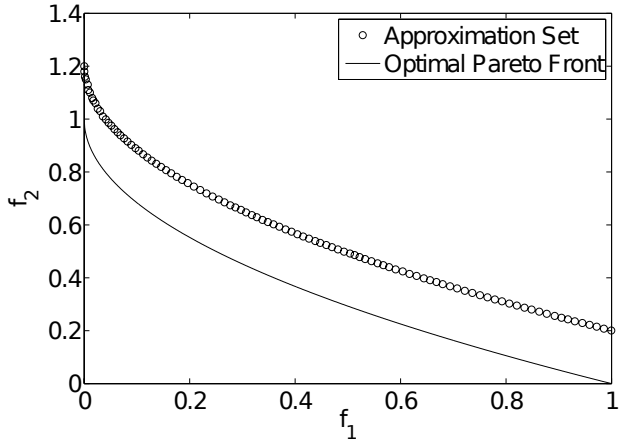
Figure 3.6 shows different fronts depicting the concepts of convergence and diversity. The uppermost front shows an example of good convergence but poor diversity: the approximation set contains Pareto optimal solutions but there are some unexplored regions of the optimal front. The approximation set depicted in the middle illustrates poor convergence but good diversity: it has a diverse set of solutions but they are not Pareto optimal. Finally, the lowermost front depicts an approximation front with both good convergence and good diversity.

3.3.3 Design issues

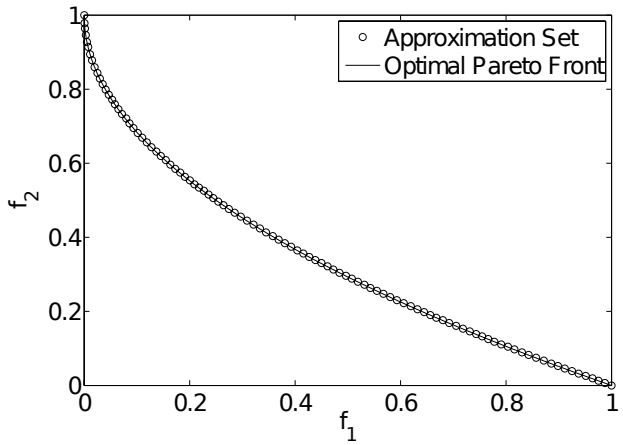
The use of Pareto optimality based techniques means, on the one hand, dealing with a set of non-dominated solutions, which requires some specific mechanisms to handle them, and, on the other hand, finding a set of Pareto optimal solutions which must be diverse enough to cover the whole front. Although depending on the algorithm there are many different issues to cope with, the following ones are commonly found in many of the existing techniques: fitness function, diversity management, and constraint handling mechanisms. Next, we discuss these points.



(a)



(b)



(c)

Figure 3.6: Examples of Pareto fronts. From top to bottom: (a) good convergence and bad diversity, (b) bad convergence and good diversity, and (c) good convergence and diversity.

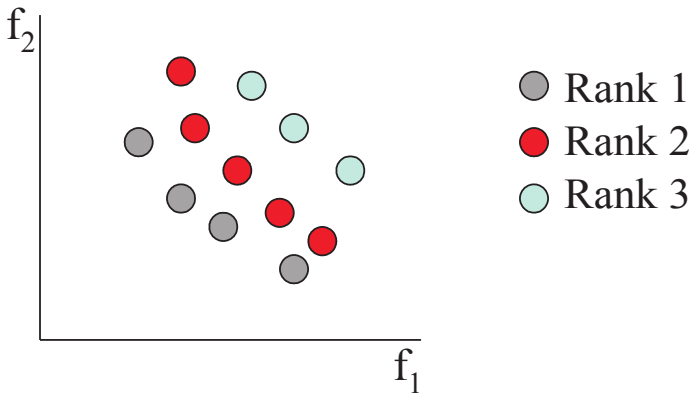


Figure 3.7: Example of sorting (*ranking*) of solutions in a bi-objective MOP.

Fitness function

In the life-cycle of any metaheuristic technique there always exists a phase in which all the solutions must be sorted to pick one (or more) of them. Examples of these phases are the selection and replacement mechanisms in EAs, or the reference set updating procedure in scatter search algorithms. In single-objective optimization, the fitness is a single (scalar) value, and thus, the sorting is done according to it. However, in the multi-objective field, the fitness consists of a vector of values (one value per objective function), and as a consequence, the sorting is not straightforward.

The dominance relationship (Definition 9) is the key issue in Pareto optimality based techniques since it allows us to sort all the solutions. Actually, this relation defines a partial order relationship, since it is not reflexive, symmetric, but an anti-symmetric, transitive relationship. Thus, different methods have been proposed in the literature ([47, 57]), which basically transform the fitness vector into a single value. Actually, this kind of strategy was first proposed by Goldberg in [86] for guiding a GA population towards the Pareto front of a given MOP. The basic idea behind it consists in successively finding solutions that are non dominated by other solutions (the best ones according to the dominance relationship). The highest possible value is assigned to those solutions. Then, the next fitness value is assigned to the solutions that become non-dominated after the previous ones are removed from the population. The procedure continues until there is no solution left in the population. Figure 3.7 depicts an example of the behavior of this sorting mechanism (where f_1 and f_2 are the objective functions which should be minimized). This strategy is known as *ranking*.

The above described procedure is the most basic one. Other advanced schemes, such as the *strength* of SPEA2 ([235]), take into account the number of solutions dominating each other as well.

Diversity management

Even though the Pareto dominance based fitness function guides the search towards the Pareto front, this approximation is not enough when a MOP is tackled. As we mentioned in Section 3.3.2, besides convergence, we seek for diversity in the front for it to be useful to the decision maker.

Although different approximations exist in the literature ([47]), many of the state-of-the-art ones are based on complementing the dominance based fitness function with a density estimator, which measures the crowd around a solution inside the objective space. Thus, given two solutions with the same fitness function value (*ranking*, *strength*), the density estimator discriminates between them attending to their diversity. Let's consider the set of solutions in Figure 3.8. In this figure, solution 1 can be considered as the best one regarding the density of solutions since it is in the less crowded area. On the other hand, solution 3 is the worst one due to it being surrounded by many other close solutions. Some well-known density

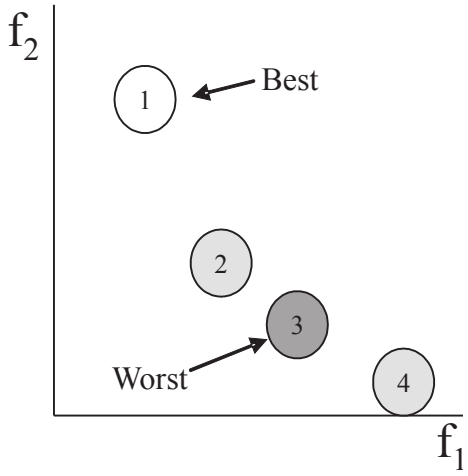


Figure 3.8: Density estimator example for non-dominated solutions in a bi-objective MOP.

estimators are: *niching* of MOGA ([79]) and NSGA ([197]), the adaptive grid of PAES ([118]), *crowding* of NSGA-II ([59]), and the k-nearest neighbor distance of SPEA2 ([235]).

Constraints handling mechanism

The MOP definition (Equation 8) presented in Section 3.3.1 explicitly includes constraints, as they are present in the typical scenario when considering real world problems, such as the ones tackled in this thesis. Constraints can be divided into two types: hard or weak constraints. A constraint is said to be hard when it should be satisfied in order for a solution to be acceptable. Meanwhile, a constraint is weak when it can be relaxed somehow in order for a solution to be accepted.

In multi-objective optimization, the scheme used by most of the state-of-the-art metaheuristics consists in considering that feasible solutions (those which do not violate any constraint) are better than non-feasible ones, regardless of their objective values ([56, 57]). Thus, given two solutions there are three possible cases:

1. If both solutions are feasible, the fitness function explained in Section 3.3.3 should be used to distinguish between them; in case of being non-dominated (they have the same fitness value), a density estimator must be applied.

2. If only one of them is a feasible solution, it should be considered as the best one.

3. If both solution are infeasible, the one which less violates the constraints is considered to be the best.

Finally, it is important to explain how to measure the amount of restriction violation by a solution. The most used scheme in the literature consist in transforming all the restriction to *greater-or-equal-than* zero type: $g_i(\vec{x}) \geq 0$, according to the MOP definition (Equation 8, [57]). This can be considered as a kind of normalization, in such a way that the value $g_i(\vec{x})$ is considered to measure the constraint violation. The main drawback of this strategy is produced by the equality restriction $h_i(\vec{x}) = 0$. For weak constraints, it can be relaxed to $h_i(\vec{x}) \geq 0$. However, when dealing with hard constraints the transformation is not that easy (specially with non-linear restrictions). As shown in [55], it is possible to convert those hard equality restrictions into weak ones carrying a loss of precision. This is an important result, since it allows to consider all the restrictions as being of the same type. There exist many other constraint handling mechanisms ([47, 57]), but we have only detailed the one used in this thesis.

3.4 Parallel and distributed metaheuristics

Even though the use of metaheuristics alone can significantly reduce the complexity and time length of the search process, still that time remains large for some real problems that need to be solved. With the recent development of cheap efficient platforms for parallel computation, it comes as natural to leverage on their power to accelerate the resolution process for these complex problems. There is an extensive literature on parallelization of metaheuristic techniques ([6, 50, 54, 147]), since it constitutes an interesting approach for not only reducing computation times, but also even obtaining higher performances of the solution process (i.e., solutions of higher quality). This improvement is due to a new search model that enables a finer tuning between intensity and diversity. Furthermore, many researchers use these parallel models on non parallel execution platforms for they offer better performances than their sequential counterparts.

This section serves as a general introduction for the most common parallelization techniques and issues found with metaheuristics. As such, since both trajectory based metaheuristics and population based metaheuristics have parallel models proposed in the literature, these will be presented in sections 3.4.1 and 3.4.2, respectively, for the sake of completeness, albeit only the latter is used in the work of this thesis. Additionally, a theoretical analysis of the migration properties and its effect on the convergence process of a population based parallel metaheuristic is presented in 3.4.3; this study will later serve as the basis for the proposed automatic tuning strategy for distributed GAs.

3.4.1 Parallel models for trajectory based methods

The parallelization methods for trajectory based metaheuristics found in the literature can be classified into three types: parallel execution of several methods (*multiple executions model*), parallel exploration of the neighborhood (*parallel movements model*), and parallel computation of the fitness function (*movement acceleration model*). They are briefly outlined next.

- **Multiple executions model:** this model corresponds to the parallel execution of several homogeneous or heterogeneous subalgorithms, all being trajectory based ([10, 146]). There are different possible configurations, depending on whether the subalgorithms collaborate during their execution time or not. The simplest case in which all executions are completely independent is widely used for its simplicity; in this case the parallel execution is equivalent to a set of sequential executions, but still has the advantage of the parallel execution (i.e., less total wall clock time). On the other side, in the cooperative case (right side of Figure 3.9), the different subalgorithms exchange information during their execution time. In this case the behavior of the parallel algorithm differs largely from the one of the sequential counterpart. Typical parameters that need to be set for this kind of technique include the selection method for the exchanged information, the use of the received information, and the schedule for these exchanges.

- **Parallel movements model:** trajectory based methods have to explore the neighborhood in each iteration and select a solution from it. This step is particularly costly (computationally speaking), since a full neighborhood usually contains a large number of individuals that have to be evaluated. This model aims to accelerate this process by a parallel exploration of the neighborhood (left side of Figure 3.9). Under a master-slave model, the master (which is actually running the main algorithm) transfers the current solution to every slave. Each slave explores then only a fraction of the neighborhood, then returns the most promising solution found. Among all the received solutions, the master selects one to continue the process. The behavior of the algorithm is the same as the sequential counterpart, but its execution is accelerated.

- **Movement acceleration model:** in the majority of cases, the most computationally expensive process of the optimization algorithm is the evaluation of a solution, that is, the calculation of the fitness

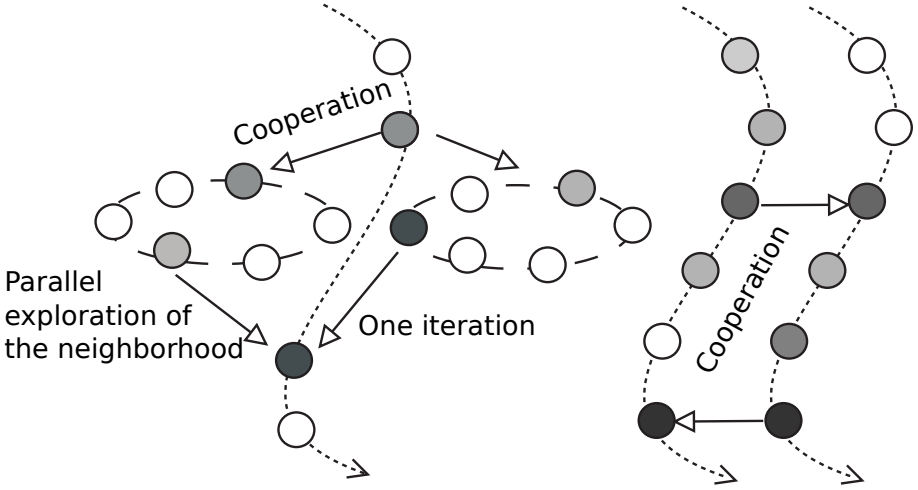


Figure 3.9: Parallel models for trajectory based methods. Left: parallel movements model, where the neighborhood is explored in parallel. Right: multiple executions model, where several cooperating subalgorithms are executed in parallel.

function. This calculation can, sometimes, be broken down into smaller independent parts that produce the global fitness by some simple combination. This model makes use of this property (when found) and has the different parts of the fitness function calculated in parallel by different processors, hence obtaining the fitness value faster. Again, this model produces the same behavior as the sequential counterpart.

3.4.2 Parallel models for population based methods

When handling populations, parallelism comes out in a natural way, as different individuals may be operated independently. Hence, the performance of population based algorithms tends to improve as they are executed in parallel. From a high level viewpoint, parallel strategies for this kind of method can be classified into two categories: (1) parallel computation, where the individual operations are performed in parallel, and (2) parallel population, where the algorithm's population is structured into smaller subpopulations.

One of the most frequently used models that follows the first strategy is the so called *master-slave* model (also known as global parallelization). Within this model, the central process –the master– performs the population-scale operations (such as the selection method of an EA), while the slaves perform the independent individual-scale operations (such as the individual fitness value computation, mutation, and sometimes the recombination as well). In this model, the global behavior of the algorithm does not diverge from the sequential counterpart, but its computation wall clock time is reduced. This kind of strategy is mostly used in scenarios where the fitness value computation is a costly process (in computation time). Another popular strategy consists in accelerating the computation time by performing multiple independent executions at a time (with no interaction among them) in that many computers; upon completion of all the executions, the best solution found among all is kept. Again, this process does not change the global behavior of the algorithm, but reduces the computation wall clock time.

Besides the master-slave model, most parallel population based algorithms found in the literature use some kind of structure for their population of individuals. This kind of model is specially used with EAs. Among the most popular models for structured populations are the *distributed model* or coarse grained, and the *cellular model* or fine grained ([12]).

In the case of distributed algorithms ([5]) (right side of Figure 3.10), the global population is divided

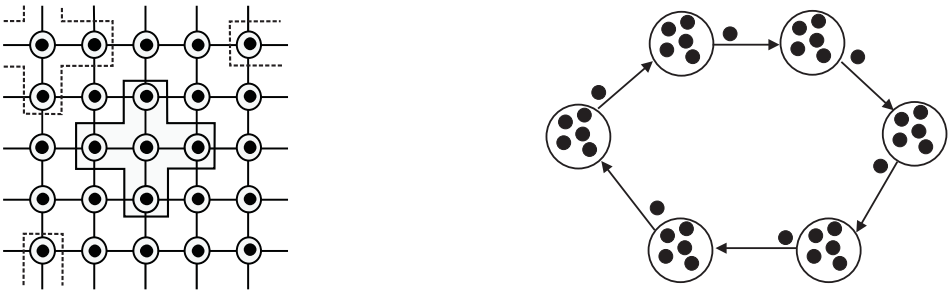


Figure 3.10: Structured population models: (left) cellular and (right) distributed.

into a set of smaller subpopulations or islands, each of which is then handled in parallel by a sequential metaheuristic. Islands cooperate by exchanging information (typically individuals); this cooperation is used to introduce new diversity into the subpopulations, keeping them from stagnating around local optima. The parameters required for the complete definition of this model include: the topology, which determines the directions of the logical communication channels among islands; the migration schedule, which determines at which moments of the execution the information exchanges will take place (since the communications are typically periodic, this parameter is normally reduced to the value of the migration period); the migration ratio, which determines the amount of information (i.e., number of individuals) exchanged; the selection and replacement criteria, which determine, in the case of migrating individuals, which individuals enter and leave each island. Finally, the communication among islands can be made to be synchronous, or asynchronous.

Alternatively, cellular metaheuristics ([65]) (left side of Figure 3.10) are based on the concept of neighborhood[1]. Each individual has a set of close individuals or *neighbors* according to some virtual superimposed regular structure (like in a crystal or a beehive) with which the exploitation of solutions will be performed. Exploration and diffusion of solutions to the rest of the population happens in a smooth fashion, due to the continuous overlap existing among the different neighborhoods, which lets high quality solutions to propagate over the population.

Besides these two basic models, there are many existing hybrid models in the literature that combine two-tiered strategies. For instance, a commonly found strategy is one in which coarse grain is used in the higher tier, and a cellular model is used within each subpopulation.

3.4.3 Theoretical analysis of the convergence in distributed EAs

In [11], the authors proposed an iterative mathematical model for calculating the growth curve of distributed genetic algorithms (dGAs) with panmictic islands. The growth curve represents the percentage of the global population that has been "occupied" by the optimal solution at any time during the execution (a single optimal solution is present in the initial population), using only selection mechanisms and reproduction. The time by which the complete population is occupied is known as "takeover time", and is a relevant value. It is based in the seminal idea that each island converges according to a logistic model, and that the entire population grows up as a sum of the growth of each component island according to the specific configuration of the migration policy (the migration topology and period are explicitly considered in the equation). Specifically, the growth $P(t)$ at instant t can be obtained as:

[1] Once more, the concept of *neighborhood* for a cellular metaheuristic is different from the ones previously mentioned for different contexts, such as trajectory based methods or particle swarm techniques.

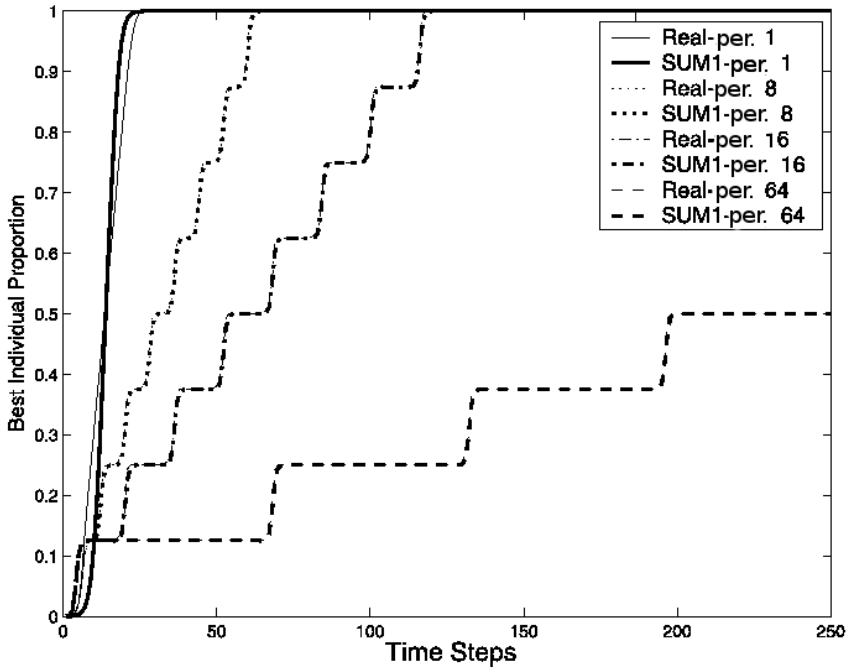


Figure 3.11: Predicted growth curves for a dGA using different migration period values (SUM), confronted against the real experimental growth curves.

$$P(t) = \sum_{i=1}^{i=d(T)} \frac{1/N}{1 + a \cdot e^{-b \cdot (t - per \cdot (i-1))}} + \frac{N - d(T)/N}{1 + a \cdot e^{-b \cdot (t - per \cdot d(T))}} \ , \tag{3.23}$$

where per is the migration period, N is the number of islands, and $d(T)$ is the length of the longest path between any two islands (known as *diameter*). This model is an extension of the logistic model proposed by Sarma and De Jong for cellular EAs ([181]). In fact, in the panmictic case ($d(T) = 0, per = 0$, and $N = 1$), this equation simplifies to the logistic one.

Later, from the growth curve equation (3.23) a closed equation for the takeover time calculation can be extracted (Eq. 3.24):

$$t^* = per \cdot d(T) - \frac{1}{b} \cdot ln \left(\frac{1}{a} \cdot \frac{\varepsilon}{N - d(T) - \varepsilon \cdot N} \right), \tag{3.24}$$

where t^* is the takeover time value, and ε is the desired level of accuracy of the mathematical model (a small value near zero).

After an experimental analysis of the growth curves and takeover regime of dGAs, the results showed how the models appropriately captured the effects of the most important parameters of the migration policy: migration period, rate, and topology (see Figure 3.11).

This model is taken as the basis for a self-adaptive migration mechanism developed for a distributed GA. The application of the model is discussed in Section 6.5.1.

3.5 Evaluation of the results

As was said before, metaheuristics are non-deterministic techniques, hence different executions of the same algorithm over the same problem instance can produce different results. This can cause inconveniences to researchers at the time of evaluating and assessing those results, and when comparing different algorithms.

Although there are works that tackle the theoretical analysis of many heuristic methods and problems ([87, 113]), this kind of theoretical analysis still involves a great deal of complexity, therefore a most commonly adopted approach is to establish the comparisons on the basis of empirical data. For this, some indicators have to be defined that enable such comparisons. In a wide sense, there are two kinds of indicators. On the one hand, there are indicators that measure the quality of the obtained solutions. Since both mono-objective and multi-objective problems are solved in this thesis work, specific indicators have to be defined for both approaches. On the other hand, there are indicators that measure the performance of the algorithms in terms of their required computation time or the amount of resources they use. Although the following discussion comments the two types of indicator separately, they are closely related and are often used together for the evaluation of metaheuristics, since the purpose of the latter is twofold: finding high quality solutions within reasonable time.

Once the indicators have been established, a given number of unrelated or independent executions of the experimental configuration (algorithmic configuration and problem instance) are required to obtain statistically consistent results. A value of 30 executions is a commonly adopted and accepted minimum, though higher values (such as 100) are recommended. The mere use of mean value and standard deviation, albeit quite frequent in the literature, is not sufficient and can lead to wrong conclusions. Thus, a global statistical analysis should be applied on the results before stating whether the observed differences are meaningful, and not just the result of the inherent randomness of the techniques.

This section contains the discussion of the indicators used in the first place (for quality and performance), then the statistical tests that are used to assess the significance of the results.

3.5.1 Quality indicators

Quality indicators or metrics are of paramount importance when evaluating a metaheuristic. They are defined in many ways depending on whether the optimal solution is known or unknown for the problem at hand (in a benchmark or a classic literature problem the optimum is often known, but for real problems this is hardly the case). As stated before, there are specific indicators for mono-objective and multi-objective problems.

Quality indicators for mono-objective problems

When the optimum is known beforehand, a simple and intuitive quality indicator for the metaheuristic is the expectancy of actually finding the optimum, or *hit rate*. This indicator is defined as the ratio or percentage of the number of executions in which the optimum is found over the total number of independent executions that have been performed. Unfortunately, knowing the optimum is not the common case for real problems or, even if they were known, sometimes they are so difficult to obtain that no execution of the experiment achieves it; in fact, experiments with metaheuristics are normally tailored to finish after a given computational effort has been spent (like visiting a maximum number of points of the search space, or running for a given time).

For these cases in which the optimum is not known in advance, or that the hit rate cannot be used, other indicators are used. The most popular are the mean and median of the best fitness value found in each independent execution. In general, other statistical data are required, such as the standard deviation, and a corresponding statistical analysis, in order to assess the statistical confidence on the observed results, should be performed.

In problems where the optimum is known, both metrics can be combined to offer a wider picture: for instance, a low hit rate with a high mean value speaks for the robustness of the method, and could be preferred over a higher hit rate but with lower median (assuming maximization).

Quality indicators for multi-objective problems

Contrary to single-objective optimization, where assessing the performance of a metaheuristic mainly requires to observe the best value yielded by an algorithm (i.e., the lower the better, in case of minimization problems), in multi-objective optimization this is not applicable. Instead, an approximation set to the optimal Pareto front of the problem is computed. As we stated in Section 3.3.2, two properties are usually required: convergence and a uniform diversity. A number of quality indicators for measuring these two criteria have been proposed in the literature: Generational Distance (GD, [205]), Inverse Generational Distance (IGD), Hypervolume (HV, [236]), Epsilon ([119]), Spread or Δ ([57]), Generalized Spread indicators, and others. Some of them are intended to measuring only the convergence or diversity, and others take into account both criteria. Figure 3.12 depicts a classification of the indicators based on which aspect they measure.

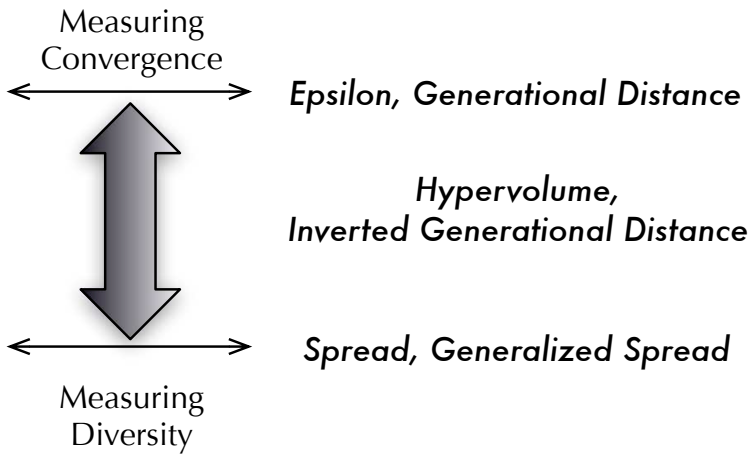


Figure 3.12: A classification of quality indicators.

- **GD**. This indicator was introduced by Van Veldhuizen and Lamont ([205]) for measuring how far the elements in the computed approximation are from those in the optimal Pareto; it is defined as:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \ , \tag{3.25}$$

  where n is the number of solutions in the approximation and d_i is the Euclidean distance (measured in objective space) between each of these solutions and the nearest member in the optimal Pareto front. A value of $GD = 0$ indicates that all the generated elements are in the Pareto front.

- **IGD**. It is a variant of the Generational Distance. It measures the distances between each solution composing the optimal Pareto front and the computed approximation. It can be defined as follows:

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \ , \tag{3.26}$$

being n the number of solutions in the optimal Pareto front and d_i is the Euclidean distance (measured in objective space) between each point of that front and the nearest member of the approximation.

- **HV**. This indicator calculates the volume, in the objective space, covered by members of a non-dominated set of solutions Q, e.g., the region enclosed into the discontinuous line in Figure 3.13, $Q = \{A, B, C\}$, for problems where all objectives are to be minimized ([236]). Mathematically, for each solution $i \in Q$, a hypercube v_i is constructed with a reference point W and the solution i as its diagonal corners. The reference point can simply be found by constructing a vector of worst objective function values. Thereafter, a union of all hypercubes is found and its hypervolume (HV) is calculated:

$$HV = \text{volume}\left(\bigcup_{i=1}^{|Q|} v_i\right). \tag{3.27}$$

Fronts with larger values of HV are desirable.

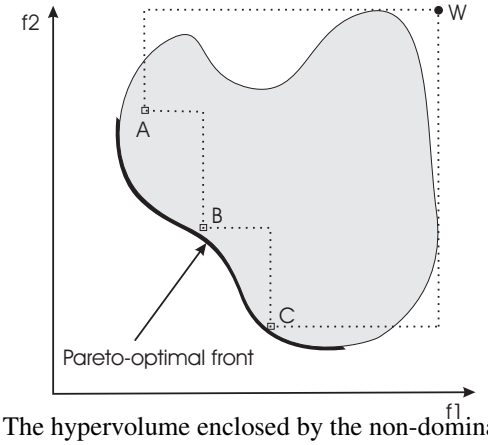


Figure 3.13: The hypervolume enclosed by the non-dominated solutions.

- **Epsilon**. Given a computed front A for a problem, this indicator is a measure of the smallest distance one would need to translate every solution in A so that it dominates the optimal Pareto front of this problem. More formally, given $\vec{z^1} = (z_1^1, \ldots, z_n^1)$ and $\vec{z^2} = (z_1^2, \ldots, z_n^2)$, where n is the number of objectives:

$$I_{\epsilon+}^1(A) = inf\left\{\epsilon \in \mathbb{R} | \forall \vec{z^2} \in \mathcal{PF}^* \; \exists \vec{z^1} \in A : \vec{z^1} \prec_\epsilon \vec{z^2}\right\}, \tag{3.28}$$

where, $\vec{z^1} \prec_\epsilon \vec{z^2}$ if and only if $\forall 1 \leq i \leq n : z_i^1 < \epsilon + z_i^2$.

- **Spread or** Δ. This indicator measures the extent of spread by the set of computed solutions. It is defined as ([57]):

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left|d_i - \bar{d}\right|}{d_f + d_l + (N-1)\bar{d}}, \tag{3.29}$$

where d_i is the Euclidean distance between consecutive solutions, $\bar{d}$ is the mean of these distances, and d_f and d_l are the Euclidean distances to the *extreme* solutions of the optimal Pareto front in the objective space (see Figure 3.14). This indicator takes a zero value for an ideal distribution, pointing out a perfect spread of the solutions in the Pareto front.

- **Generalized Spread** The previous indicator is based on calculating the distance between two consecutive solutions, which works only for 2-objective problems. This metric is extended by computing

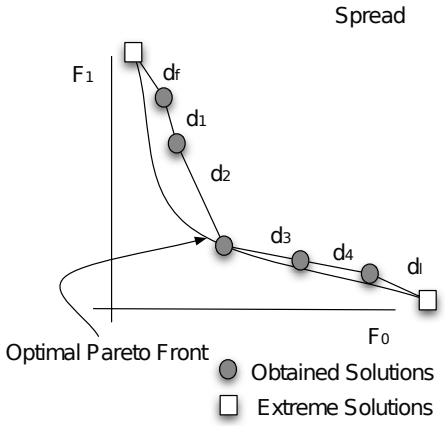


Figure 3.14: Distances from the extreme solutions.

the distance from a given point to its nearest neighbor in [168]. This extension is based on the metric proposed in [232]:

$$\Delta = \frac{\sum_{i=1}^{m} d(e_i, S) + \sum_{X \in S} \left| d(X, S) - \bar{d} \right|}{\sum_{i=1}^{m} d(e_i, S) + |S^*|\, \bar{d}} \quad , \tag{3.30}$$

where S is a set of solutions, S^* is the set of Pareto optimal solutions, $(e_1, ..., e_m)$ are m extreme solutions in S^*, m is the number of objectives and

$$d(X, S) = \min_{Y \in S, Y \neq X} ||F(X) - F(Y)||^2 \quad , \tag{3.31}$$

$$\bar{d} = \frac{1}{|S^*|} \sum_{X \in S^*} d(X, S). \tag{3.32}$$

Since those indicators are not free from arbitrary scaling of objectives, in this work they are applied always after normalizing the objective function values.

3.5.2 Performance indicators

A performance measure is one that is associated to the time or amount of computational resources used by the metaheuristic, which are usually measured as the number of visited solutions in the search space (computational effort), or the computation time. Many researchers favor the number of solution evaluations over the time to measure the computational effort, since it is impervious to implementation details, software or hardware, hence rendering comparisons independent of those factors. However, this measure can be misleading in some cases, where the evaluations are non-homogeneous with some requiring much more time than others (this is often the case in genetic programming [124]), or when the operators besides the fitness evaluation are much more costly in one technique than in some other. In general, the combined use of both metrics (number of evaluations and time) is advisable to obtain a more realistic picture of the computational effort.

Since some algorithms will run on parallel computation platforms, a brief discussion on the main indicators used in the literature for this scenario follows. Among them, the most important for parallel algorithms is the *speedup*, which compares the execution time of the sequential algorithm with the equivalent time of the parallel counterpart. The speedup is an indicator of how many times faster the parallel algorithm is with respect to the sequential one. If we note as T_m the computation time for a given algorithm running on m

processors, then the speedup is the ratio of the swiftest execution on a monoprocessor system T_1 over the execution time on m processors T_m:

$$s_m = \frac{T_1}{T_m} \ . \tag{3.33}$$

For non-deterministic algorithms this metric cannot be used directly, instead the *mean* computation times have to be compared:

$$s_m = \frac{E[T_1]}{E[T_m]} \ . \tag{3.34}$$

The main difficulty for this measure exists for the unclear significance of T_1 and T_m. In [12], a classification is made among different existing speedup measures according to the significance of these values (see Table 3.1).

Table 3.1: Speedup measure taxonomy ([12]).

I. Strong Speedup
II. Weak Speedup
A. Speedup with solution quality stopping criterion
1. Versus panmixy
2. Orthodox
B. Speedup with predefined effort

The *strong speedup* (type I) compares the execution time of the parallel algorithm to the most efficient sequential algorithm. This is the most accurate definition of speedup, but because of the difficulty of obtaining the most efficient sequential algorithm, many researchers in the field of parallel algorithms choose not to use it. The *weak speedup* (type II) compares the parallel algorithm with the equivalent sequential counterpart. In this case, two stopping criteria can be used: based on the obtained solution quality, and based on the maximum allowed effort. The last one is advised against, since it ends up comparing times of algorithms that produce different outputs (solutions not having similar quality), which defeats the purpose of this metric. Thus, two variants are proposed for weak speedup with stopping criterion based on solution quality: compare the parallel algorithm to the canonical sequential version (type II.A.1), or compare the execution time of the parallel algorithm in a processor with the time that same parallel algorithm spends on m processors, (type II.A.2). In the first case two different algorithms (sequential and parallel) are compared, while in the latter a single algorithm is compared with itself running on different platforms (single processor and m processors).

Although speedup is the most frequently used metric, there are other metrics defined to measure the behavior of a parallel algorithm. We briefly sketch two other such metrics: parallel efficiency and serial fraction.

The *parallel efficiency* (Eq. 3.35) is a normalization of the speedup over the number of processors m. It takes values between 0 and 1 indicating the degree of use of the processors used:

$$e_m = \frac{s_m}{m} \ . \tag{3.35}$$

Karp and Flatt ([111]) developed another metric to measure the performance of any parallel algorithm. This metric is called *serial fraction* of the algorithm (Eq. 3.36). The lower the value of the Karp-Flatt metric, the better the parallelization of the code.

$$f_m = \frac{1/s_m - 1/m}{1 - 1/m} \ . \tag{3.36}$$

3.5.3 Statistical analysis of the results

Having defined the indicators of quality and performance, one needs at least 30 independent executions to obtain a set of values for each indicator. From the statistics viewpoint, these data can be considered as a sample from a probability density function and, in order to extract the correct conclusions, a statistical analysis has to be performed on these results ([60, 187]).

The procedure adopted in our research work is as follows. First, a Kolmogorov-Smirnov test is performed in order to check whether the samples are distributed according to a normal distribution (Gaussian) or not. For non-normal distributions, a Kruskal-Wallis test is performed. For normal distributions, the homocedasticity (i.e., equality of variances) is checked using the Levene test. If the Levene test returns a positive value, an ANOVA test is performed; otherwise a Welch test is performed. The confidence degree for all tests is set to 95% (corresponding to a significance level of 5% or a *p*-value below 0.05). Figure 3.15 graphically sketches this process.

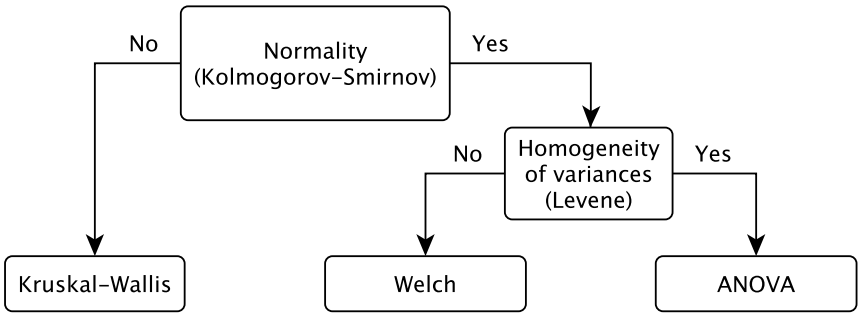


Figure 3.15: Statistical analysis process of the experimental results.

Since typically more than two algorithms are involved, a post-hoc testing phase which allows for multiple comparison of samples ([93]) is also performed. Specifically, the `multcompare` function provided by Matlab © is the one used, since it is capable of selecting the most adequate critical value depending on the sample. This function uses tests ranging from the most conservative ones, as *HSD* or *Tukey-Kramer*, to the less conservative ones, like *Scheffe's S* test. The confidence level is kept at the same value ($\alpha = 0.05$).

Chapter 4

Algorithms

This chapter serves as a general introduction to the metaheuristic techniques that are used throughout this thesis work to solve the different optimization problems selected. Only generic descriptions of the algorithms will be given here, "templates" of the high-level behavior of the algorithms. The specific implementation details (like the operators for mutation or crossover), which are problem-specific (or representation-specific), are delayed to the corresponding chapters where the application of the algorithms to solve the problems is discussed. We first describe the mono-objective techniques used in Section 4.1, then describe the multi-objective techniques used in Section 4.2.

4.1 Mono-objective techniques

In this thesis, mono-objective approaches are tackled for the Radio Network Design problem (chapters 5 and 6) and the Location Discovery problem (chapters 9 and 10). The algorithms used for this kind of approach, and described in this section, are Simulated Annealing (Section 4.1.1), Genetic Algorithm (Section 4.1.3), CHC (Section 4.1.2), and Particle Swarm Optimization (Section 4.1.4).

4.1.1 Simulated Annealing

Simulated Annealing (SA) is a trajectory based optimization technique ([21]). It was first proposed by Kirkpatrick et al. in [117]. SA is a fairly commonly used algorithm that provides good results and constitutes an interesting method to compare to other optimizing methods because of its simplicity. The pseudocode for this algorithm is shown in Algorithm 1.

The algorithm works iteratively keeping a single tentative or candidate solution s_a at any time. In every iteration, a neighbor solution s_n is generated, which either replaces or not the current solution depending on an acceptance criterion. The acceptance criterion works as follows: both the old (s_a) and the new (s_n) solutions have associated quality values (*fitness*); if the new solution has better fitness than the current solution, it replaces the current solution. Otherwise, the replacement is done with probability P, which depends on the difference between their quality values and a control parameter T (*temperature*). This acceptance criterion provides a way of escaping local optima. The mathematical expression for the probability P is shown in Equation 4.1.

$$P = \frac{2}{1 + e^{\frac{fitness(s_a) - fitness(s_n)}{T}}}. \tag{4.1}$$

The temperature parameter is reduced during the search process following a given cooling schedule. We employ the geometric rule $T(n+1) = \alpha \cdot T(n)$, with $0 < \alpha < 1$, performed every k iterations (k is

Algorithm 1 Pseudocode of SA.

```
1: t ← 0;
2: Initialize(T,s_a)
3: Evaluate(s_a)
4: while not EndCondition(t,s_a) do
5:    while not CoolingCondition(t) do
6:       s_n ← ChooseNeighbor(s_a)
7:       Evaluate(s_n)
8:       if Accept(s_a,s_n,T) then
9:          s_a ← s_n
10:      end if
11:      t ← t + 1
12:   end while
13:   Cooldown(T)
14: end while
```

the *Markov chain length*). This makes SA accept only better solutions towards the end of the search. The initial temperature $T(0)$ is set to a value such that starting from a random solution, SA will accept the first neighbor with probability 80%.

A mutation operator is used to produce s_n from s_a. Mutation operators used in SA are described in sections 6.2.2 and 10.2.2.

4.1.2 CHC

The Cross-generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation (CHC), is a kind of Evolutionary Algorithm (EA) that was first proposed by Eshelman in [69]. Like other EAs, CHC works with a set of solutions (*population*) at any time. The pseudocode for this algorithm is shown in Algorithm 2.

Algorithm 2 Pseudocode of CHC.

```
1: t ← 0
2: Initialize(P_a,convergence count)
3: while not EndingCondition(t,P_a) do
4:    Parents ← SelectionParents(P_a)
5:    Offspring ← HUX(Parents)
6:    Evaluate(Offspring)
7:    P_n ← ElitistSelection(Offspring,P_a)
8:    if not Modified(P_a,P_n) then
9:       convergence count ← convergence count-1
10:      if convergence count == 0 then
11:         P_n ← Restart(P_a)
12:         Initialize(convergence count)
13:      end if
14:   end if
15:   t ← t + 1
16:   P_a ← P_n
17: end while
```

In every step, a new set of solutions is produced by selecting pairs of solutions from the parent population P_a and recombining them. An incest prevention criterion prevents individuals that are too similar to each other to mate, and recombination is made using a special procedure known as HUX. This procedure

copies first the parents into the offspring, then randomly exchanges half of the diverging information between the offspring. This method has been designed to preserve the maximum amount of diversity in the population, which is an important matter since no new diversity is introduced during the iteration (because there is no mutation operator). The next population is formed by selecting the best individuals among the parent and the offspring populations (elitism).

In a normal execution, population convergence is sooner or later achieved, thus the previously described algorithmic behavior would stall on it. For this reason, a special mechanism is used to introduce new diversity when this happens: the *restart* mechanism. Upon restarting, all the solutions except the very best ones (or only *the* best one) are significantly modified through a high rate mutation (typically $p_m = 0.35$).

The HUX and the mutation used in the restart mechanism are detailed in Section 6.2.2.

4.1.3 GA

Genetic Algorithms (GAs) also belong to the wide family of EAs ([16]). They appear for the first time as a widely recognized optimization method as a result of the work of John Holland in the early 70's, and particularly his 1975 book. The pseudocode for this algorithm is shown in Algorithm 3.

A standard GA is a population based technique ([21]) that uses a selection operator to pick solutions from the population (line 4), a crossover and a mutation operators to produce new solutions from them (lines 5-6), and a replacement operator to choose the individuals for the next population (line 8).

Algorithm 3 Pseudocode of GA.

```
1: t ← 0
2: Initialize(P_a)
3: while not EndingCondition(t,P_a) do
4:    Parents ← SelectionParents(P_a)
5:    Offspring ← Crossover(Parents)
6:    Offspring ← Mutate(Offspring)
7:    Evaluate(Offspring)
8:    P_n ← Replacement(Offspring,P_a)
9:    t ← t + 1
10:   P_a ← P_n
11: end while
```

Our implementation of the genetic algorithm typically uses a ranking method for parent selection and elitist replacement for the next population, that is, the best individual of the current population is included in the next one. Should different operators be used, they will be explicitly described in the corresponding section.

The mutation and crossover operators used with GA are described in sections 6.2.2 and 10.2.2.

Distributed Genetic Algorithm

We use a parallel GA that implements the distributed model (coarse grained) presented in Section 3.4.2. We shall refer to this algorithm as distributed Genetic Algorithm (dGA). In our dGA, each island executes as a GA (Algorithm 3), with an additional step at the end of the inner loop: at some special iterations an inter-island communication called migration takes place, during which each island sends an individual from its population to the *next* island[1], and receives an individual from its *preceding* island. Thus, apart from the migration-specific ones, the operators used in dGA are the same used in GA: crossover and mutation, which are described in Section 6.2.2.

[1] According to the topology of the dGA.

Our implementation of the dGA uses a unidirectional ring topology (see Figure 3.10), and synchronous migration. The global population of the algorithm is the union of the subpopulations hosted at the different islands.

4.1.4 Particle Swarm Optimization

PSO is a population based metaheuristic inspired in the social behavior of birds within a flock. In a PSO algorithm, each potential solution to the problem is called *particle* and the population of solutions is called *swarm* (hence the name of the algorithm). The best values visited so far for each solution, p_{best}, as well as the best value visited so far by any particle of the swarm, g_{best}, are stored.

Algorithm 4 Pseudocode of PSO.

```
1: InitializeSwarm(S, p_best)
2: g_best ← LocateLeader(S)
3: t ← 0
4: while not EndingCondition(t) do
5:    for all p_i in S do
6:       p_i ← UpdatePosition(p_i, p^i_best, g_best)
7:       Evaluate(p_i)
8:       p^i_best ← UpdatePbest(p_i, p^i_best)
9:    end for
10:   g_best ← UpdateLeader(S)
11:   t ← t + 1
12: end while
```

Algorithm 4 describes the pseudo-code of a general single-objective PSO. The algorithm starts by initializing the swarm (line 1), which includes both the positions and speeds of the particles. The corresponding p_{best} of each particle is initialized, as well as the leader (line 2). Then, during a predefined number of iterations, each particle *flies* through the search space (updates its position, line 6), is evaluated (line 7), and its p_{best} is calculated (lines 6-8). At the end of each iteration, the leader is updated. Besides, as the execution progresses, the inertia weight linearly evolves from an initial value to a final value (which is generally lower).

The flight operator used in PSO is described in Section 10.2.2.

4.2 Multi-objective techniques

In this thesis, multi-objective formulations are defined for the Radio Network Design problem (chapters 5 and 6) and the Wireless Sensor Network Layout problem (chapters 7 and 8). The algorithms used for this kind of approach, and described in this section, are NSGA-II (Section 4.2.1), PAES (Section 4.2.2), SPEA2 (Section 4.2.3), MOCell (Section 4.2.4), and MOCHC (Section 4.2.5).

4.2.1 NSGA-II

Deb et al. proposed in [59] the second Nondominated Sorting Genetic Algorithm (NSGA-II) as a multi-objective technique that dealt with the main problems existing in the field: high computational complexity of nondominated sorting, lack of elitism and need of a sharing parameter specification. The authors fixed these problems by using a fast non-dominated sorting, an elitist Pareto dominance selection and a crowding distance method.

NSGA-II is based on a genetic algorithm. Its behavior can be seen in Algorithm 5. The differences between this algorithm and mono-objective GAs lie within the fitness assignment strategy.

Algorithm 5 Pseudocode of NSGA-II.

```
1: t ← 0
2: Initialize(P_a)
3: while not EndingCondition(t,P_a) do
4:    Parents ← SelectionParents(P_a)
5:    Offspring ← Crossover(Parents)
6:    Offspring ← Mutate(Offspring)
7:    P_i ← Merge(P_a,Offspring)
8:    RankingCrowding(P_i)
9:    P_n ← ElitistSelection(P_i)
10:   t ← t + 1
11:   P_a ← P_n
12: end while
```

In NSGA-II, the solutions are first sorted according to restriction fulfillment. Feasible solutions come first, then unfeasible solutions are sorted by increasing degree of constraint violation. Feasible solutions and every set of solutions with the same violation degree are then respectively sorted according to Pareto dominance. This sorting is performed by successively extracting from the chosen subpopulation the current set of non-dominated solutions (fronts). All the solutions in a front are given the same rank value, beginning at 0 for the first front extracted, 1 for the second, and so on. This way, solutions can be sorted according to rank, starting at 0. Finally, within every group of solutions having the same rank, solutions are sorted according to the *crowding distance*. This criterion places first those solutions whose closest neighbors are farthest, thus enhancing diversity.

The mutation and crossover operators used with NSGA-II are described in sections 6.2.2 and 8.2.2.

4.2.2 PAES

The Pareto Archived Evolutionary Strategy (PAES) is a multi-objective evolutionary strategy that does not use self-adaptation, or recombination (crossover). Hence, PAES is a trajectory-based technique. Despite PAES handles a single solution at a time, a full Pareto optimal set is required as the execution's output; to generate such a set, PAES uses an external *archive* in which non-dominated solutions found are stored, and returns that archive upon execution completion. Algorithm 6 sketches the operation of PAES.

This algorithm maintains a single solution, and mutates it at each iteration to generate a new candidate solution (line 5). This new candidate solution either replaces the current one or not, and either enters the archive or not, subject to a Pareto-dominance criterion (lines 8, 10, 12, 15). Since the archive has bounded size, not all non-dominated solutions may be stored; a criterion based on the distribution of solutions over the front determines which solutions are accepted into the archive. Specifically, PAES employs a diversity measure based on an adaptive grid to uniformly distribute the non-dominated solutions in the front.

The mutation operator employed with PAES is described in Section 8.2.2.

4.2.3 SPEA2

The Strength Pareto Evolutionary Algorithm (SPEA2) is a multi-objective evolutionary algorithm. SPEA2 was proposed by Zitler et al. in [235]. We show the algorithm's pseudocode in Algorithm 7.

SPEA2 uses a population and an archive simultaneously in its operation. In it, each individual is assigned a fitness value that is the sum of its strength raw fitness and a density estimation. The strength value S of a solution i represents the number of solutions (in either the population or the archive) that are dominated by that solution, that is, $S(i) = |\{j | j \in P_t \cup \overline{P_t} \wedge i > j\}|$. The strength raw fitness value R of a given solution i, on the contrary, is the sum of strengths of all the solutions that dominate it, and is subject to minimization, that is, $R(i) = \sum_{j \in P_t \cup \overline{P_t}, j > i} S(j)$. The algorithm applies the selection, crossover, and

Algorithm 6 Pseudocode of PAES.

```
1: Archive ← ∅
2: Initialize(c)
3: t ← 0
4: Insert(Archive, c)
5: while not EndingCondition(t) do
6:    m ← Mutate(c)
7:    Evaluate(m)
8:    if Dominate(m,c) then
9:       Discard (m)
10:   else if Dominate(c,m) then
11:      Insert(Archive, m)
12:   else if Dominate(Archive,m) then
13:      Discard(m)
14:   else
15:      Test(c, m, Archive)
16:   end if
17:   t ← t + 1
18: end while
```

Algorithm 7 Pseudocode of SPEA2.

```
1: t ← 0
2: Initialize(P_0, P̄_0)
3: while not EndingCondition(t,P̄_t) do
4:    FitnessAssignment(P_t, P̄_t)
5:    P̄_{t+1} ← NonDominated(P_t ∪ P̄_{t+1})
6:    if |P̄_{t+1}| > N̄ then
7:       P̄_{t+1} ← Truncate(P̄_{t+1})
8:    else
9:       P̄_{t+1} ← FillWithDominated(P̄_t)
10:   end if
11:   Parents ← BinaryTournament(P̄_{t+1})
12:   Offspring ← Crossover(Parents)
13:   P̄_{t+1} ← Mutate(Offspring)
14:   t ← t + 1
15: end while
```

mutation operators to fill an archive of individuals; then, the nondominated individuals of both the original population and the archive are copied into a new population. If the number of non-dominated individuals is greater than the population size, a truncation operator based on calculating the distances to the k-th nearest neighbor is used (a typical value is $k = 1$), $D(i) = \frac{1}{\sigma_i^k + 2}$, where σ_i^k is the distance from solution i to its k-th nearest neighbor. This way, the individuals having the minimum distance to any other individual are chosen.

4.2.4 MOCell

MOCell is a recent proposal based on the cellular model (that structures the population of solutions), for multi-objective optimization. Algorithm 8 shows its pseudocode.

MOCell first creates an empty Pareto front (line 2). The individuals are placed in a 2D toroidal grid, and undergo the reproductive cycle iteratively (lines 5 to 14) until the stopping condition is met (line 4). This way, for each individual, the algorithm selects two parents, each through a binary tournament, one is taken

Algorithm 8 Pseudocode of MOCell.

```
1: Initialize(P)
2: ParetoFront ← CreateEmptyFront()
3: t ← 0
4: while not EndingCondition(t,P) do
5:    for all i in P do
6:       Neighbors ← GetNeighborhood(i)
7:       Parents ← Selection(Neighbors, ParetoFront)
8:       Offspring ← Recombination(Parents)
9:       Offspring ← Mutate(Offspring)
10:      Evaluate(Offspring)
11:      Insert(P,i,Offspring)
12:      InsertParetoFront(i, ParetoFront)
13:   end for
14:   t ← t + 1
15: end while
```

from the grid neighborhood, and the other from the external archive. The winner of each tournament is determined by its crowding distance inside the neighborhood and the archive, respectively. The selection of a parent from the archive introduces front solutions (intensity), thus guiding the search towards promising regions. The selected parents are then recombined, and the resulting offspring is mutated and evaluated. This newly produced individual is then inserted in the population, replacing the solution in the current neighborhood with the worst crowding distance. The new individual may be inserted in the external archive as well, using a similar procedure as in PAES, but with NSGA-II's crowding distance as the diversity measure instead of the adaptive grid.

MOCell can also handle restrictions in the problem in the same way NSGA-II does. When comparing two solutions, if both are feasible, Pareto-dominance is used. If only one is feasible, this one dominates the other. When none is feasible, the one with the less restriction violation dominates the other.

The recombination and mutation operators used by MOCell are described in Section 8.2.2.

4.2.5 MOCHC

The multi-objective version of the CHC algorithm, namely MOCHC, maintains the basic structure of the algorithm shown in Algorithm 2, and uses the same crossover and re-initialization mechanisms. The differences between the two techniques lie in two aspects: the selection mechanism and the restart procedure.

- **Selection**: In CHC, an elitist selection that sorts solutions based on their fitness values is used, but fitness is no longer used in a multi-objective approach. In MOCHC, the solutions of the merged population (including both parents and offspring) are sorted according to their *ranking* and *crowding distance* estimators, similar to those used in NSGA-II. Thus, the non-dominated solutions in the population are selected and removed from it, constituting the subset of rank 0. This process is iteratively repeated, producing the subsets of rank 1, 2, 3, and so on, until the number of solutions extracted is equal or greater than the population size; in the second case, the crowding estimator is applied to the solutions of the last subset extracted and those with the higher distance values are selected.

- **Restart**: In single-objective CHC only the solution with the highest fitness is kept, in multi-objective the interest is to keep information concerning the best set of solutions found, not just the best solution found. Thus, instead of a single solution, a small percentage of the population is kept. The percentage of solutions kept corresponds to the solutions with higher ranking and crowding distances. The percentage was empirically chosen to be 5%.

4.3 Conclusions

In this chapter we have presented and described the different optimization algorithms that are employed throughout this thesis work to solve the different problems addressed. We have structured the chapter in two sections: the mono-objective techniques are described in the first section, and the multi-objective techniques are described in the second section. In the mono-objective domain we present the trajectory-based algorithm SA, two evolutionary algorithms, GA and CHC, and the particle-based algorithm PSO. In the multi-objective domain we present the trajectory-based algorithm PAES, three evolutionary algorithms, NSGA-II, SPEA2 and MOCHC, and a cellular algorithm, MOCell.

Part II

RADIO NETWORK DESIGN

Chapter 5

Radio Network Design Problem

In this chapter we present the first problem tackled in this thesis, namely the Radio Network Design problem, or RND ([30, 33]). In this problem, which can be found in the telecommunications field, a subset of location sites has to be selected from a set of available locations. A Base Station Transceiver (BTs or BS) will be placed in each of the selected sites, offering coverage to a terrain area called *cell*, so that ideally, the resulting network offers radio coverage to a given area.

RND is closely related to the design process followed for cellular networks, such as the access networks used nowadays in the major systems for mobile telephony (GSM, UMTS). Withing the context of mobile telephony, an RND-like problem can be found under the name of Automatic Cell Planning (ACP, [160]), more specifically, the *site selection* of ACP. The ACP problem, together with the Automatic Frequency Planning problem (AFP, [1, 121]), constitute the main tasks the service provider needs to solve when an access network is undergoing its design phase. Both RND's and ACP's goals are the selection of locations to place Base Station Transceivers (BTs or BS), or equivalently the terrain partition into cells (since the cell location and boundary depends on the location of its BS, both problems amount to the same decisions). Later, AFP's goal is to assign frequency sets to the different cells in order to maximize the communication bandwidth and minimize the interferences among different channels.

Specifically, in RND the global coverage offered by the union of the coverage cells of the BSs placed has to be maximized, while the number of such BSs has to be kept to a minimum (for economic efficiency). The problem instances can have varying dimension, depending on whether different types of BS antenna can be chosen, and whether those antennae have parameters that have to be tuned; but at its core, RND remains an NP-hard combinatorial problem.

An important problem exists in the domain of Wireless Sensor Networks that, by its nature and definition, has a strong similarity with the RND problem considered here: we refer to that problem as the scheduling problem ([195]). We will describe that problem in this chapter, and point out the similarities and the differences between RND and scheduling. Additionally, , we will sketch indications as to how to extend the existing solutions for RND in order to obtain solutions for scheduling. In the next chapter we address the discussion of the solutions for RN.

5.1 Problem description

Mobile communications are a major area in the telecommunications industry of the twenty-first century. They require the use of a mobile device by the end user, the presence of a network accessible by the mobile device from any place the user has to be. Among many different techniques and applications, two of the best known paradigms are ad-hoc networks and access networks; in the first, the very mobile terminals form the communication network in an ad-hoc fashion (hence its name), while in the latter, a fixed infrastructure

offers a radio access over a given geographic area, which is called the covered or coverage area.

A generally accepted method for building an access network is to discretize the terrain to be covered using some points that represent terrain areas, each of which can be covered by a transmitter conveniently located in some base station (BS); this solution is known as a radio network. Figure 5.1 shows the architecture of the GSM network, which is a well-known cellular network. In GSM, each cell is covered by an antenna corresponding to a site, which is controlled from a base station controller (BSC), which in turn is connected to one of the mobile switching centers (MSC) that form the internal backbone. The system uses databases for authentication of users (AuC), home and visitor locations (HLR and VLR) to monitor the location of the users as they roam through the network, The short message service center (SMSC), flexible number register (FNR), and service data and service control points (SDP and SCP), complete the system.

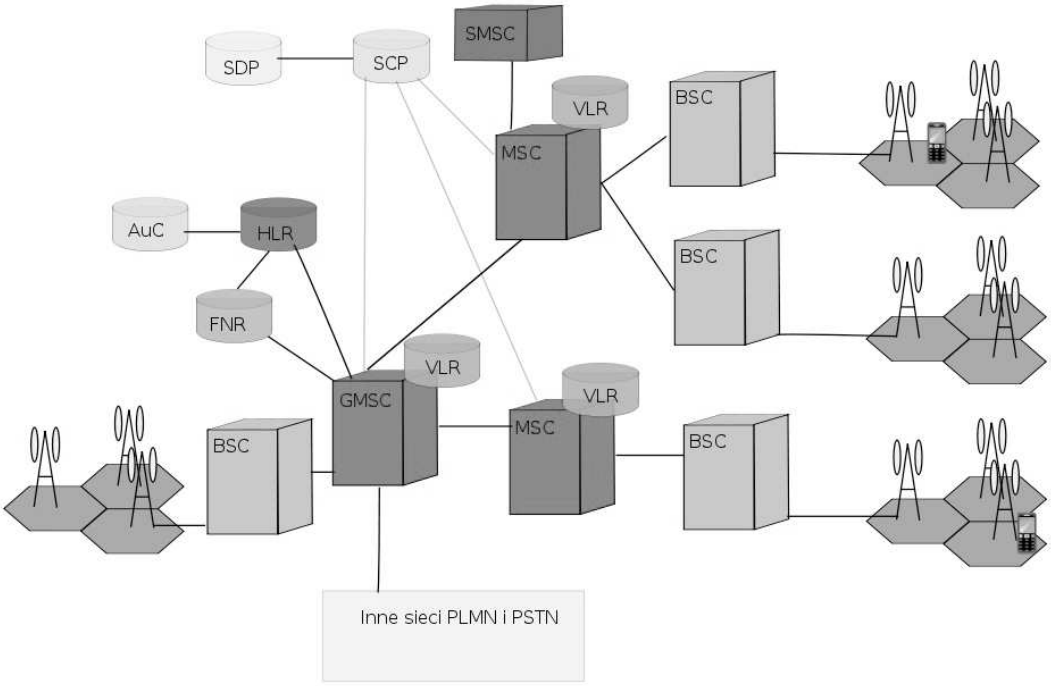


Figure 5.1: Architecture of the GSM cellular network.

Presently, a number of companies that have entered and are well established in the sector, compete to obtain the highest number of clients by offering the best service at the lowest cost. As a result, there is an increasing interest in solving the inherent optimization problems involved in the design process of the service infrastructure –in the case of RND, the network of base stations that provide the radio access of the cellular network.

Thus, the problem we solve is how to achieve maximum coverage of the terrain in order to obtain a valuable service for the customer (ideally the coverage should be complete), while placing the lowest number of transmitters, so that the cost of the service remains competitive. This is equivalent to selecting the optimal positions for placing the transmitters, and this problem is known as the Radio Network Design problem (RND), or the radio coverage problem.

The part of an area that is covered by a transmitter is called a *cell*. In the following we will assume that the cells and the area considered are discretized, that is, they can be described as a finite collection of

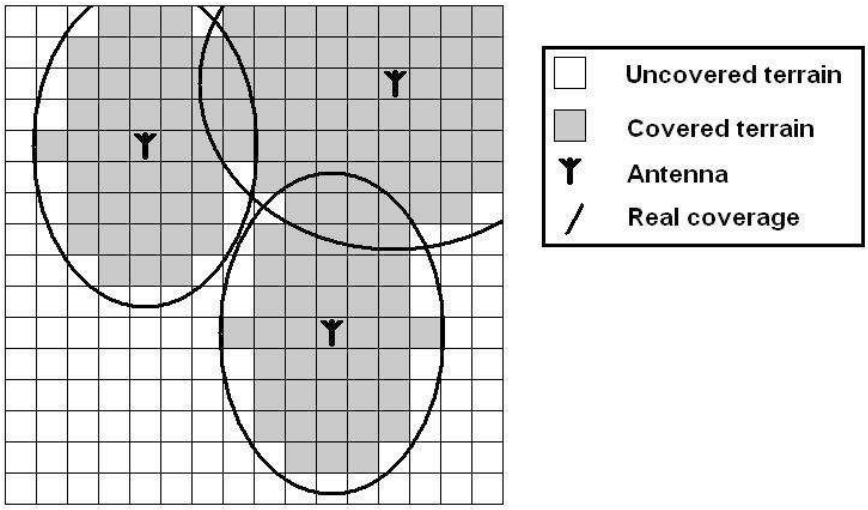


Figure 5.2: Three candidate transmitter locations and their associated covered cells on a grid.

geographic locations. The computation of cells may be based on sophisticated wave propagation models, on measurements, or on draft estimations. In any case, we assume that cells can be computed and returned by an *ad hoc* function.

A formal definition of the basic RND is as follows. Let us consider the set L of all potentially covered locations and the set M of all potential transmitter locations. Let G be the graph $(M \cup L, E)$, where E is a set of edges such that each transmitter location is linked to the locations it covers and let $x \subseteq M$ be a solution to the problem, where each element of x is a site selected to install a BS.

Searching for the minimum subset of transmitters that covers a maximum surface of an area comes to searching for a subset $M' \subseteq M$ such that $|M'|$ is minimum and such that $|Neighbors(M', E)|$ is maximum, where

$$Neighbors(M', E) = \{u \in L \mid \exists v \in M', (u, v) \in E\}. \tag{5.1}$$

The problem we consider recalls the unicost set covering problem (USCP) that is known to be NP-hard. The radio coverage problem differs, however, from the USCP in that the goal is to select a subset of transmitters that ensures a *good* coverage of the area and not to ensure a *total* coverage. The difficulty of our problem arises from the fact that the goal is twofold, no part being secondary. If minimizing the number of sites was the primary goal, the solution would be trivial: $M' = \emptyset$. If maximizing the number of covered locations was the primary goal, then the problem would be reduced to the USCP.

Throughout this chapter we will consider other versions of the RND problem, which differ in the type of antennae that might be placed in every location. The simple versions, as the one described above, use antennae that require no parameters to determine their coverage. The more complex versions use antennae that require some configuration parameters (i.e., direction) to determine the cell area they cover.

5.2 Coverage models in Radio Network Design

For a complete definition of a RND problem instance, the following elements need to be defined:

- List of available location sites (ALS, M in the previous definition of the problem).
- Available antenna types and their properties (cost, parameters, etc.).

- Coverage offered by each antenna type at each of the available location sites.

When the RND problem is going to be solved, an initial study of the terrain has to be carried. In this study, a set of available locations for the base stations is chosen, and the coverage offered by a BS in each of those sites is obtained. The criteria used to select the initial list of available location sites is very varied and may include good signal transmission/reception properties, ease of construction/maintenance, economic reasons, health reasons (avoid hospitals and schools), among others ([179, 180]). The characterization of the coverage obtained by antennae in each of the sites is often obtained by on-site measurements with specific devices. Both the obtaining of the ALS and the characterization of the coverage properties of the sites contained are beyond the scope of our work; hereafter we consider that both have been obtained and are inputs to the problem at hand.

Setting aside the ALS definition, which defines the instance properties but does not affect the nature of the problem, the amount of combinations for the nature of RND is still boundless. Therefore, we propose a coarse classification of existing RND problem conceptions based on the approach adopted for the calculation of the coverage: the test points model, and the regular grid model.

5.2.1 Test points model

The first approach adopted for the evaluation of the coverage in the RND problem is the use of set of test points ([13]). When this approach is adopted, the problem is generally referred to as Automatic Cell Planning (ACP). The coverage is then represented by the coverage properties obtained at these points; the points need therefore to be carefully selected in order to provide a faithful representation of a continuous area coverage, however this is also beyond our scope.

The model of test points generally classifies the points used in several levels, according to the use they will have. A popular implementation of this model adopts a three-level system:

- At the lowest level are the reception test points (RTP), $\mathcal{R} = \{R_i / i \in [1, \ldots, l]\}$, $R_i \in \mathbb{R}^3$, where l is the number of such RTPs. These points are used to measure the intensity or strength of the received radio signal.

- At a higher tier are the service test points (STP), $\mathcal{ST} = \{ST_i / i \in [1, \ldots, k]\}$, where k represents the maximum number of such STPs. These points are selected from the wider set of $\mathcal{R}$, and are used to measure the quality of service; generally, a minimum threshold has to be surpassed to guarantee acceptable performance of the system.

- At the highest level are the traffic test points (TTP), $\mathcal{T} = \{TT_i / i \in [1, \ldots, n]\}$, where n is the number of such TTPs. These points, selected from the wider set of $\mathcal{ST}$, are used to model the traffic generated by users in the system, hence each has an associated expected traffic volume e_i.

Figure 5.3 shows a graphical example of the relationships among the three types of test points, as an equivalent area relation of the existing set relation: $\mathcal{T} \subseteq \mathcal{ST} \subseteq \mathcal{R}$.

The use of test points has the advantage of representing the coverage (and general service) properties through a reduced and manageable set of values. Hence, test points can be combined with empirical on-the-field coverage measurements –despite the high cost of the latter– to determine the different cells resulting from placing antennae at different sites of the ALS. The cell of an antenna is the number of STP that receive a strong enough signal from that antenna; the information obtained from the measurements is often coded as a matrix of transmission gains from the locations of the ALS to the points in STP (the larger matrix with all points in RTP is also available, but not used because of the larger memory requirements it would involve).

In this definition of the problem, the coverage may be considered as a constraint or as an optimization objective. In the first case, the problem definition may require that some percentage of or all STP receive

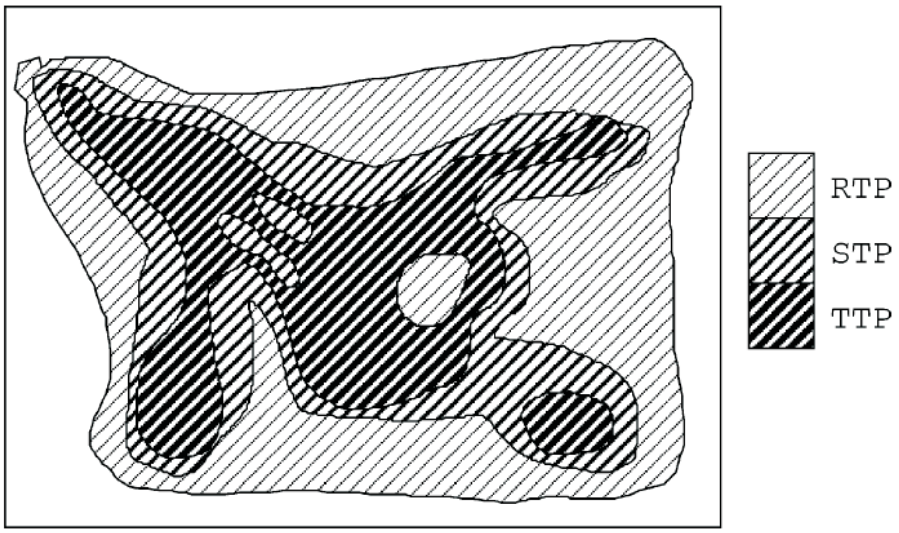


Figure 5.3: Area relation between reception test points (RTP), service test points (STP), and traffic test points (TTP).

sufficient coverage. In the second case, a possible calculation of the coverage in this case is the following (Equation 5.2).

$$Coverage(\vec{x}) = \frac{1}{|\mathcal{ST}|} \sum_{i \in \{1,\ldots,|\mathcal{ST}|\}} \max_{j \in \{1,\ldots,|\mathcal{M}|\}} \{x_j \cdot C_{ij}\}, \tag{5.2}$$

where

$$x_j = \begin{cases} 1 & \text{if an antenna is placed in the location } m_j \\ 0 & \text{otherwise} \end{cases},$$

and we assume the coverage matrix C defined as

$$C_{ij} = \begin{cases} 1 & \text{if an antenna placed in the location } m_j \text{ covers the service test point } ST_i \\ 0 & \text{otherwise} \end{cases}.$$

Additionally, ACP can have up to two more objectives defined: reduce the interferences from foreign BTSs (the calculation is again handled in matrix form) and guarantee that each TTP receives sufficient bandwidth to serve all the traffic requirements (each BTS has a traffic capability, the BTSs covering any TTP have to provide it with larger traffic capability than its estimated traffic). These objectives are not found in the canonical definition of RND.

5.2.2 Regular grid model

An alternative system to evaluate the coverage of a system (not limited to a cellular system), is by a regular terrain discretization into a grid ([33]). The regular grid model can be viewed as a chessboard-like division of the terrain into tiles or area points. The tiles form a regular geometrical structure that equal the complete terrain; triangular, rectangular of hexagonal grids can be used in this sense.

Both the test points and the grid models make use of discrete points to represent portions of terrain; the main difference between the two is that in the test points model there is a high flexibility for choosing the points, while in the grid model they must fit into some regular lattice. As a result, the number of points in the grid is much larger than the number of test points. Besides, in the test points model there is an explicit hierarchy among the points (as the one with three levels described in the previous section), while in the grid model all points are in principle of equal importance (although more advanced grid models can use a *differentiated* coverage criterion, requiring different coverage degrees in different points, or even imposing the coverage of some points as a constraint).

The large number of points involved in the grid model renders the use of empirical on-the-field measurements intractable due to its prohibitive cost (there are too many points, it is practically impossible to take measurements in all of them for all the sites in the ALS). Therefore, the commonly adopted approach in this scenario is to use a wave propagation model. This approach can be taken to different complexities depending on the types and parameters of the antennae, and the properties of the soil; despite this, the wave propagation model is often reduced to a homogeneous case: the geometrical shape of the cell defined by the region (collection of grid points) where the received signal strength surpasses the defined threshold. The cell shape is then independent from the location (that is, centered at the site, but not varying from one site to another), and marks the points that are covered by the antenna.

The grid model with the geometrical cell shape is quite popular. Figure 5.4 shows three example cell shapes (the shaded areas indicate the corresponding covered areas), with different degrees of realism: the squared cell shape, conceptually simple but far from reality, the omnidirectional or circular cell shape, with corresponds to an ideal isotropic wave propagation model, and the directional or sectorized cell shape, which is the most accurate model. For the first two models the only possible parameter in the antenna is the transmission power, the third adds two new parameters: direction and angular width.

Since the shape of the coverage is independent from the position of the base station (space-invariant), the coverage can be defined as a set of vectors $Cov = \{\overrightarrow{cov}_i\}$ such that for any two points p_a, p_b in the grid $\mathcal{G}$, we have that p_a covers p_b if and only if $\overrightarrow{p_b p_a} \in Cov$. Then the calculation of the coverage for the case of the terrain grid is straightforward (see Equation 5.3).

$$Coverage(\vec{x}) = \frac{1}{|\mathcal{G}|} \sum_{i \in \{1,\ldots,|\mathcal{G}|\}} \max_{j \in \{1,\ldots,|\mathcal{M}|\}} \{x_j \cdot c_{ij}\}, \tag{5.3}$$

where

$$x_j = \begin{cases} 1 & \text{if an antenna is placed in the location } m_j \\ 0 & \text{otherwise} \end{cases},$$

and we use the coverage indicator c_{ij} defined as

$$c_{ij} = \begin{cases} 1 & \text{if } \overrightarrow{m_j p_i} \in Cov \\ 0 & \text{otherwise} \end{cases}.$$

In our definition of RND, we use the **regular grid point model** for the estimation of the coverage.

5.3 Literature review for the RND problem

The RND problem has received much attention in research. We briefly review the main approaches in optimization to solve different variations of RND in the literature in this section.

Watanabe, Hiroyasu, and Mikiand ([210]) work out a parallel evolutionary multi-objective approach for deciding the antennae placement and configuration in cellular networks. The authors present two parallel models for multi-objective GAs applied to the problem: the Master-Slave with Local Cultivation Genetic Algorithm (MSLC) and the Divided Range Multi-Objective Genetic Algorithm (DRMOGA). The MSLC algorithm is based on the standard master-slave approach, but the evolutionary operators are carried out on

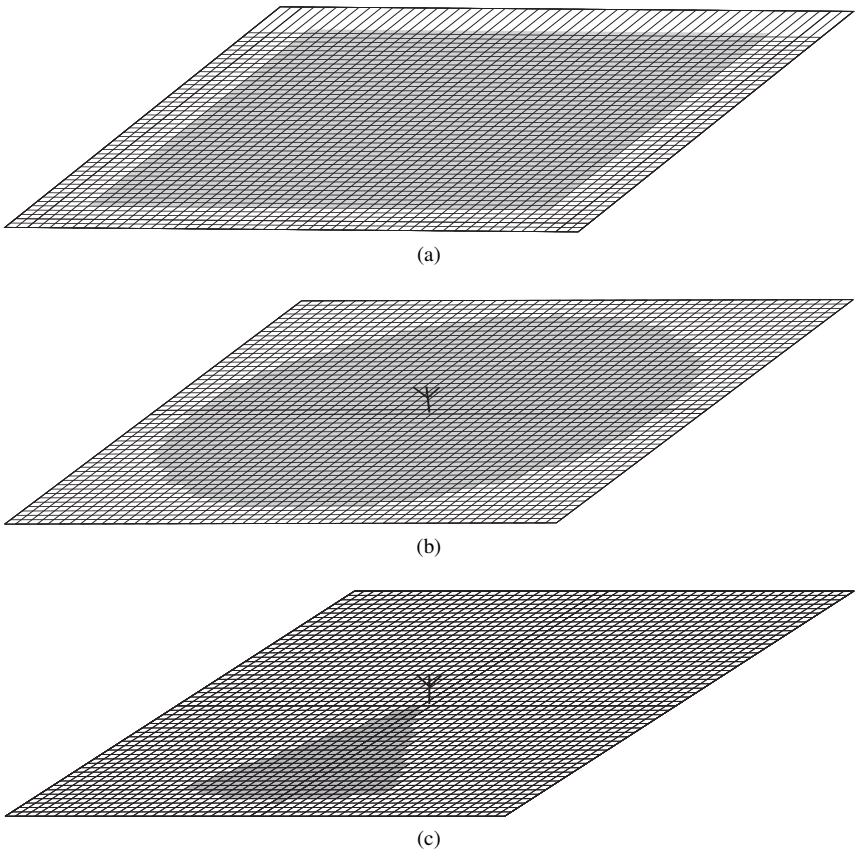


Figure 5.4: Different models employed for coverage with grid terrain: (a) squared cell, (b) circular cell, (c) sectorial cell.

the slaves using a two-individual population and the evolution follows the minimal generation gap model. DRMOGA is a standard distributed island model that uses domain decomposition. The empirical analysis compares both models proposed with MOGA ([78]) and a standard distributed GA. They show that MSLC gets the best results of Pareto front covering and non-dominated individuals, while establishing that DRMOGA results are affected by the number of subpopulations: the number of non-dominated individuals decreases when the number of subpopulations grows.

In the same line of work, Meunier, Talbi, and Reininger ([158]) present parallel implementation of a GA with a multilevel encoding deciding the activation of sites, the number and type of antennae, and the parameters of each base station. Two modified versions of the classical genetic operators, named geographic crossover and multilevel mutation, are introduced. The fitness evaluation utilizes a ranking function, similar to Fonseca and Fleming's MOGA algorithm ([78]), and a sharing technique is employed to preserve diversity among solutions. In addition, a linear penalization model is used to handle the constraint considered (a minimal value for the covered area). A master-slave parallel implementation is presented for solving high dimension problems in reasonable times, with each slave processing a part of the geographic working area. The algorithm is evaluated with a large and realistic highway area generated by France Telecom. The authors analyze the convenience of using the sharing strategy proposed instead of concentrating on a small part of the Pareto front, showing that a better Pareto front sampling is obtained in the first case.

In a later work, Cahon, Melab, and Talbi ([28]) solve the same problem with a multi-objective GA. They use the three parallel/distributed GA models implemented in the ParadisEO (PARAllel and DIStributed Evolving Objects) framework: the *island (a)synchronous cooperative* model, the *parallel evaluation of the solution* model, and the *distributed evaluation of a single solution* model ([29]). Working on a cluster of 40 Pentium III PCs, the Pareto fronts obtained for the test instances studied confirm the robustness and efficiency of the island model for solving the problem. In addition, since the fitness evaluation process demands a high computational effort, the problem is suitable for applying the parallel and distributed evaluation models. The computational efficiency analysis shows that the parallel evaluation model follows almost-linear speedup behavior. The distributed evaluation model scales super-linearly up to 10 processors, and then it follows a logarithmic decay.

Calégari et al. ([30, 33]) develop a distributed GA to find the optimal placement of antennae. In [32] the authors compare a greedy technique, a Darwinian algorithm, and a PGA. The PGA uses a bit string representation for codifying the whole set of possible antenna locations and a parametric fitness function evaluating the covered area as a function of a parameter that can be tuned in order to obtain acceptable service ratio values. Experiments are conducted on two real-life cases: Vosges (rural scenario) and Geneva (urban scenario). On average, the PGA and the greedy technique show the same solution quality. But when an optimal solution is known, it can be found using PGA whereas the greedy approach usually falls in bad attractive local optima. Alba et al. ([9]) tackle the same problem with sequential and parallel GAs over an artificial instance. In a later work, Alba and Chicano ([8]) perform a deep study on the parallel approach evaluating the influence of the number of possible locations, the number of processors, and the migration rates. They find a sublinear speedup and conclude that the isolation of the subpopulations is beneficial for the search.

Maple, Guo, and Zhang ([152]) use a PGA for solving a network planning problem, consisting in determining the optimum placement for base stations in third generation mobile networks. They propose a multi-objective approach, employing several objective functions for considering multiple network design factors. The model evaluates the network capacity (attempting to maximize the maximum number of users permitted in a cell), considers intra-cell and inter-cells interference, uses known propagation models for coverage (attempting to maximize the covering radius of a cell), and incorporates the design cost calculation (attempting to minimize the base placement cost). Using a common strategy in telecommunication network operation, the authors employ a binary representation that selects a subset of sites for base stations from a finite set of possible locations. When a site is selected for use, the GA is also used to determine the antenna height and its transmission power. For dealing with the massive solution space and the complex fitness calculation process, they propose a parallel GA following the coarse-grain subpopulation model.

The authors do not present numerical results for the optimization problem, stating that the research was "currently being undertaken" to implement the algorithm on a 30 node Beowulf cluster.

Créput et al. ([51]) propose a parallel evolutionary strategy for dimensioning a cellular network to cover a city, addressing the problem of evaluating the optimal number and location of base stations needed for satisfying QoS and traffic requirements. They use a geometric approach for facing the adaptive meshing (AM) process, where a pattern of regular hexagonal cells transform themselves and adapt their shapes according to traffic density, geometric constraints, and other parameters. For solving the problem with relatively low computational effort, the authors propose the Hybrid Islands Evolutionary Strategy (HIES), combining a hill-climbing local search procedure with a subpopulation distributed evolutionary mechanism. A high-level crossover-and-mutation schema and an elitist selection operator are used for avoiding local minima reached in the local search. In the particular approach presented in the article, each subpopulation or island is limited to contain only one individual, so the HIES proposal is similar to memetic algorithms ([164]) incorporating a geographic isolation distribution for individuals, like in a cellular PGA. All three HIES operators (local search, crossover and macromutation) are stochastic procedures, specifically designed for the problem to solve. The AM is an intrinsically multi-objective problem. However, Créput et al. use a linear aggregative fitness function considering the objective (minimization of the total number of base stations) and four constraints related to the resource distribution optimization, the regularity of cell geometry, the number of visible cells, and the elimination of overloaded cells. In the experimental evaluation, the authors consider four test instances, including a real-life scenario (city of Lyon, France) and three problems specifically built for representing typical application cases. Results show satisfactory meshing patterns, producing well-contoured meshes on a map while eliminating overloaded cells. The authors work with several values for the population size parameter ($5<$*population size*<80) in their experiments, but they do not distribute the algorithm on several machines or in a multiprocessor computer. They refer to their distributed algorithm as the *parallel version*. It has the ability of HIES to improve its performance as population size increases. The authors state that it is able to achieve highly adapted individuals using a moderate number of generations. Although the parallel version increases the population size, it allows obtaining better results than versions using lower population sizes, using a similar number of function evaluations. These results suggest that there is room to improve the HIES computational efficiency, executing on a multiple machine cluster, given that performing the simulations required from 5 to 20 hours of execution time for the test scenarios studied.

Yun and Hyun-Meen ([104]) focus their work on the parameterization of the base stations in a given area. They propose an iterative algorithm to solve the corresponding optimization problem. This algorithm partitions the problem geographically according to domain areas, then iteratively solves the resulting subproblems using a genetic algorithm. A signal to noise criterion (determined by free-space propagation model) is employed to calculate the coverage regardless of signal interference, for the sake of simplicity. Their technique is tested against a global genetic algorithm for the whole problem (without partition) and a random search technique, outperforming both.

The radio network design problem for UMTS is studied by Amaldi, Capone and Malucelli in [13]. The third generation for mobile telecommunications requires a different approach since its features allow for more flexibility in its use. The cell capacity is not limited *a priori* -resources are shared all over the network- and the main limitation is interference, therefore a capacity study had to be made. Hata's propagation model is used to deal with real-world like instances over a rectangular service area where a set S of candidate sites is defined and another set TP of test points is randomly determined. Two kinds of power control mechanisms are considered (power-based and SIR-based), and two kinds of optimization techniques are employed: greedy procedures (direct and reverse) and Tabu Search (TS). The experimental results show that TS performs better than greedy procedures, although the differences were not big.

As it can be seen, metaheuristics have been heavily used to solve the RND problem. Indeed, most of the existing work applies some version of the Genetic Algorithm to solve the problem. Multi-objective and parallel approaches are commonplace for this problem as well. Although being a versatile a robust technique, the GA can sometimes be "too generic", and therefore fail to achieve a high performance; we

propose the use of CHC, an algorithm more suited to the specific features of the RND problem, and compare its performance against a GA and a Simulated Annealing (SA) for a wide set of problem instances using different antenna models and different problem approaches. We will also consider multi-objective and parallel approaches to the problem; as a matter of fact, in a second part, we propose a novel self tuning migration operator to be used with a parallel GA to achieve high performances without requiring to tune the migration parameters.

5.4 Relationship with Wireless Sensor Networks

We finally establish the link between the RND problem described in this chapter, and the domain of WSNs. There is one important problem in WSN that, by its nature, the pursued objectives, and the data structures employed, greatly resembles RND: the node scheduling problem. We refer by this to the *sleep* scheduling problem which decides activity and sleep time of the nodes in order to maximize the network lifetime ([195]), as opposed to the *activity* scheduling problem ([76]), where the purpose is to reduce the activity cycle (i.e, the total time required by the network to have its –conflicting– nodes perform one time their scheduled tasks).

5.4.1 Scheduling problem definition

The scheduling problem is a combinatorial optimization problem defined within WSNs. The goal of this problem is to assign a working schedule to every node in the network, such that at any point in time there is a subset of the nodes in the network that is active or working; when a node is not active it is in a low energy-consumption state known as *sleep* state, or is said to be sleeping. The scheduling solution has to fulfill two objectives:

1. Optimize the quality of service offered by the network (or guarantee a minimum).
2. Maximize the network lifetime.

As a result, every node as a *duty cycle*, which is the portion of the continuous network operation time in which the node is active. Thus, the basic solution to a scheduling problem is a list of time intervals for each of the nodes, the time intervals representing when said node will be active. The key idea is to exploit the fact that the network contains an excess of nodes, that is, not all nodes need to be active for the network to achieve the required coverage. Hence, some nodes can be put to sleep while the network still produces an acceptable quality of service. If the set of active nodes is rotated through the network lifetime, then every node will be working for some time, and sleeping during the rest; this way, nodes can function for longer periods than their initial energies would allow them if they were working at full regime.

We now provide more specific definitions of the optimization objectives. In our considered domain of application of a WSN, the main purpose of the network is to monitor a given terrain field, therefore the quality of service is defined based on the degree of monitoring achieved by the network ([207]). The monitoring is measured as a coverage offered by the network; that is, the monitoring of the WSN corresponds to the area that is covered by the network. In this sense, the notion of coverage is similar to the one used in the RND problem (sections 5.1 and 5.2); thus both the test points model and the regular grid model can be used. In addition, there is one specific coverage model defined for special instances of WSN: the barrier coverage; in this case the WSN has to monitor not the inside of a terrain, but its perimeter, this kind of coverage is defined for intrusion-detection systems. Coverage issues are explored in further detail in Chapter 7; for the time being, we restrict our conception of the scheduling problem to the scenarios where it can be assimilated to the RND problem.

Unlike RND, the scheduling problem corresponds to a dynamic network, with nodes entering and leaving the active state at different moments, thus producing different network configurations and coverage

levels as a result. Therefore, the coverage is not constant, but varying in time. As a result, the quality of service metric needs some way of handling the time variations in the coverage. Popular techniques are:

- Define a minimum guaranteed value for any time.
- Use the time-average coverage.

The first option is to add a constraint, this option is the one used when the quality of service is not a maximization objective but a requirement. The second one is combined with the quality of service being an optimization objective. In the following, we will consider a quality of service defined to be a constraint.

Regarding the network lifetime, there are also many definitions for it. The most popular are the *Time To First Failure* (TTFF), and the α*-lifetime* ([229]). This will be described in further detail in Chapter 7. For the purpose of scheduling, nodes are generally considered to be spending energy at a given constant rate while they are working, and the energy they consume while sleeping is neglected. Hence, for the time being, the conception of scheduling required for scheduling relies on the following principle:

Proposition 1. *The longest the maximum node duty cycle, the shortest the resulting lifetime.*

Thus the objective of the scheduling problem becomes the minimization of the nodes duty cycles.

A common approach to solving the scheduling consists in dividing the operation time in frames, formed of (regular) periods or slots. During each slot, a given subset of nodes will be active, while the rest will be sleeping. A node is either active during the whole slot, or sleeping during the whole slot. A node can only be active during a limited number of slots per frame (typically just one). We have then the following property:

Proposition 2. *The more slots in a schedule, the shortest the maximum node duty cycle.*

Thus, scheduling amounts to obtaining the maximum number of slots, that is, the maximum number of disjoint subsets in the network such that the quality of service is fulfilled by each of the subsets. This definition of the scheduling problem can be associated to the disjoint set covering problem, which is an NP-complete problem ([34, 195]).

A formal definition for the scheduling problem setting the coverage as a constraint (instead of an optimization objective) may be as follows. Let S be a deployed WSN. Maximize k, such that there is some $\{S_1, S_2...S_k\}$, with $S_i \subseteq S$ for all i, that verifies:

$$\forall i, Coverage(\cup_{n \in S_i} n) \geq C_{min}, \tag{5.4}$$

$$\forall n \in S, \sum_{i=1}^{k} I(n \in S_i) \leq L_{max}, \tag{5.5}$$

where $Coverage : \{S\} \rightarrow \mathbb{R}$ is a function that gives the coverage degree for a given set of nodes, and the constant values C_{min} and L_{max} are the minimal coverage required and maximal working cycle of any node (typically $L_{max} = 1$), respectively. The function I is an indicator that returns 1 when the input is true and 0 otherwise. In short, the first condition states that the minimum coverage has to be achieved, and the second condition states that any node of the WSN may belong to at most L_{max} different subsets, therefore ensuring a lifetime expansion of at least k/L_{max}.

Extending RND solutions to solve scheduling

The RND problem corresponds then to a reduced scheduling problem where a single subset is extracted from the network. Thus, a simple iterative extension of RND can be used to solve scheduling, as shown in Figure 5.5. At each iteration of RND, a set of locations is extracted, corresponding to a color of the graph-coloring version of scheduling. Then, RND is solved again with the previously selected locations removed from the set of available locations, N, in order to avoid a same node being selected more than once. When no solution is found for RND, the process is finished, and the set of RND solutions S is returned as the solution to scheduling.

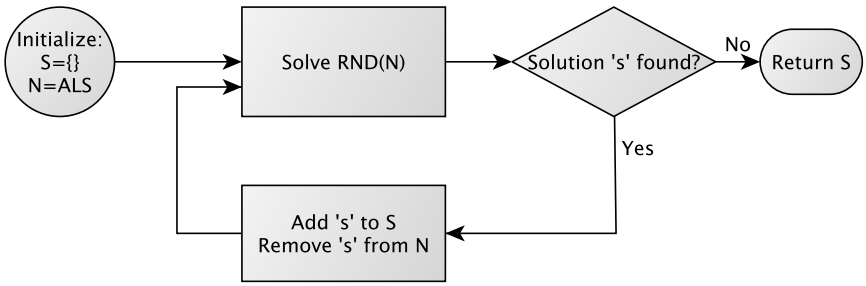


Figure 5.5: Iterative resolution of RND to solve scheduling.

5.4.2 Literature review for scheduling

Much work exists for the different definitions of the scheduling problem in sensor networks. In most of the problem approaches, the extension of the network's lifetime is the main driving force behind the resolution of scheduling.

A theoretical study on the effects of scheduling is shown in [229], where the authors focus on the α-lifetime, in networks with uniform node distribution. The number of nodes required to extend the network's lifetime to reach T times the lifetime of a single node is obtained. The concept of α-lifetime is interpreted in two different manners: in the first, the network is supposed to offer as much coverage as possible at any time (initially 1-coverage), the time lapse until it can no longer offer more than α-coverage is the lifetime ($\alpha \leq 1$ is the ratio of achieved coverage to the maximum feasible coverage); in the second, the network offers α-coverage at any time, the moment it fails to do so is the lifetime.

Many works design the scheduling strategy with the objective of ensuring coverage. The most common conception of coverage is *area coverage*. A centralized approach is presented in [195], where the authors extract the maximal number of subsets from the WSN such that each subset of nodes offers maximal coverage of the region. The authors consider the intersection points between coverage boundaries of the deployed nodes and identify the "critical" areas as the regions covered by few nodes; then they propose a heuristic algorithm that avoids that two nodes covering the same critical area belong to the same subset, hence maximizing the number of subsets. However, distributed techniques are much more popular. For instance in [219], the authors present the Coverage Configuration Protocol (CCP), an algorithm for distributed scheduling of the nodes in a WSN that guarantees k-coverage. In CCP, the nodes check their coverage regions; whenever a node detects that there is a spot within its coverage region that is not k-covered by its active neighbors, it sets itself active. At first, connectivity is not an issue, since it is considered granted by the $R_{COMM} > 2R_{SENS}$ property (by which coverage guarantees connectivity). Later, they relax this assumption and guarantee coverage by the combined used of CCP with SPAN. Another distributed scheduling technique for coverage preservation is studied in [95]. The authors propose a round based system, where nodes select a new schedule at the beginning of each round. To do so, they first randomly select a time point within the round time length. A node, say node a, needs to keep a list of all its covered locations, and of all other nodes (among a's neighbors) that cover those same locations. For each location, a finds the closest preceding and succeeding nodes b and c (those with selected time points closest to it) that cover it; then a sets a time frame from the middle point between b's and a's time points to the middle point between a's and c's time points. The node a will then stay active during the union of all time frames of all its covered locations. This way, full coverage is guaranteed to every location in the network covered by at least one node. The same approach is used in [224] with differentiated coverage; the required coverage degree is used as a multiplying factor to the point's covering nodes. Additionally, the recalculation of the longest duty cycle, and the use of M different schedules in a rotation fashion, are proposed as improvements.

Two distributed scheduling protocols are presented in [96], to guarantee both coverage and connectivity in non-homogeneous WSN. If the perimeter of the coverage region of a node is k-covered by its neighbors, and all neighbors are within communication range, then k-coverage and k-connectivity are provided. In the first proposed protocol, each node independently decides to sleep at a random time after checking its neighborhood. The second protocol manages transmission power control to modify the communication radius of the node; each node checks whether it can reduce its transmission power to disconnect its farthest neighbor without violating the coverage and connectivity constraint. Finally, a combination of the two is proposed in which every node either performs the scheduling part or the transmission power control part. The combination of scheduling with transmission power and sensing power control is also studied in [234], for the general scenario where the WSN requires k_1-coverage and k_2-connectivity. Centralized and distributed techniques based on Voronoi and greedy are proposed. In the Voronoi-based technique, each node selects its sensing radius to cover its Voronoi cell, and its communication radius so that all its neighbors are connected; then, a node are only put to sleep only if its neighbors can cover its Voronoi cell, and are still connected (the may increase their radii). The general case is handled by using k_1-Voronoi and k_2-Relative Neighborhood Graph. For the greedy algorithm, the terrain is first divided into patches delimited by the nodes concentric sensing and coverage disks (with discretized radii); the algorithm iteratively proceeds by performing at each step either the activation of a new node (which must be within communication range of the already active nodes), or the increase of the radius of an active node, whichever "adds" the most terrain patches. The scheduling problem to maximize coverage by directional sensors is studied in [2]. The coverage offered by the sensors is a "pizza-portion" (like the one in Figure 5.4c), hence both the active nodes and the direction of their coverage beams have to be selected. A fitness function is defined that subtracts the number of active nodes (weighted by a constant ρ) from the coverage achieved, and an ILP solution is proposed; NP-completeness of the Maximum Coverage with Minimum Sensor (MCMS) is demonstrated. Then a greedy algorithm is presented with a centralized (CGA) and distributed (DGA) versions; the centralized iteratively selects the node and direction that maximizes the number of covered targets, while in the distributed every node has a priority value and each target is considered covered by the highest priority node that covers it, nodes select the direction in which the cover the maximal number of targets. An iterative version of the DGA, the Sensing Neighborhood Cooperative Sleeping (SNCS), is proposed. SNCS updates the selected nodes periodically, assigning priority based on the node's remaining energy. The effects of the sensing radius and the beam width, as well as the robustness to errors in the node's locations, the angles or data transmissions are assessed.

A convenient tool for the estimation of area coverage that is frequently used with scheduling is Voronoi's diagram, as has been commented for [234]. Another example is the approach proposed in [207] for the scheduling problem. The authors define a threshold such that no node with an associated Voronoi cell smaller than the threshold will be active by the end. The algorithm iteratively searches the node with smallest Voronoi cell, and if the cell is smaller than the threshold, the node is disconnected and the Voronoi diagram is recomputed for the next iteration. Globally, the algorithm finds a single dominating set based solely on coverage (like in RND). Several possibilities for a distributed implementation are also discussed. We can also notice the work in [52], where the authors present a distributed Voronoi-based scheduling procedure, RSE. Coverage is guaranteed, but connectivity is overlooked by assuming $R_{COMM} > 2R_{SENS}$. The technique uses 2-Voronoi diagram for each node to determine whether the node is superfluous and can be turned off (the 2-Voronoi diagram of a node is the Voronoi graph of the node's neighbors excluding the node itself); if both all vertexes of the 2-Voronoi diagram that fall inside the nodes cover region and the intersections of that cover region with edges of the 2-Voronoi diagram are covered by other nodes, then the node is redundant. The technique is able to handle non-homogeneous WSN by using weighted Voronoi diagrams (with weights corresponding to the sensing radii). The Voronoi diagrams are calculated in a distributed manner. The appearance of a new node, as well as the failure of a node are considered and can be handled.

Other approaches for the coverage are also taken into consideration for scheduling. In [42] a scheduling is designed to minimize the coverage breach, which is a complementary view of the coverage, that is, to

minimize the uncovered region. The authors use a point coverage model with N nodes and M points, then generate K subsets of maximum cardinality W (this is called bandwidth and is the limit to the number of active nodes at a time in the WSN) such that the accumulated breach (the number of uncovered points in any subset) is minimized. Two methods are proposed: a linear programming relaxation, followed by a greedy integer rounding, and a reverse greedy, that starts from a network with all nodes active, then iteratively removes the one with minimal coverage breach. In [43], the authors define the scheduling problem for minimal breach. The authors consider three kinds of breach: the aggregated uncovered region, the maximum uncovered region at any moment, and the maximum time length a point is uncovered; NP-completeness is demonstrated for the three problems. A bandwidth limits the maximum number of active nodes simultaneously. Two resolution methods, a greedy algorithm and an LP relaxation (where the node active status is a continuous value between 0 and 1, which will be set to either 0 or 1 at the end), are proposed. The considered WSN is a 1-hop network, thus connectivity is not an issue.

A different conception of the coverage is offered in [159] for the scheduling: target tracking coverage. Besides, this version of the problem does not aim to maximize the lifetime of the system, but to offer the best tracking performance, by assigning nodes to mobile targets. The network is assumed to have N nodes, and the time is divided into K intervals; there are M mobile targets in the network. Each target requires one sensor to track in each time interval, the same sensor cannot track two different targets at a time. A Modified Particle Swarm Algorithm (MPSO) is proposed to solve the scheduling problem in order to maximize the number of completed tasks (tracked targets), and the mean tracking accuracy. The modified version of PSO uses $M \times K$ integers in $[0; N]$ matrices to represent the nodes assigned to each target at each interval, and modified operators to ensure that all modified solutions are feasible.

Other works do not consider a measure of coverage as their main objective; for instance in [90], the nodes are evaluated based on the correlation between their sensed data, instead of the coverage. Both centralized and distributed algorithms are proposed for the scheduling problem in this scenario. According to the authors's definition, a set of nodes is correlated to another node if the latter's sensed data can be reconstructed with bounded error by a linear combination of the former's. Connectivity is also taken into account: the selected set of nodes is only valid if it is fully connected with the HECN. The distributed algorithm needs local detection of correlation among nodes; nodes have a self-assigned priority value and enter sleep mode based on than priority, and the state of their neighborhood. When a node enters sleep state, it sends a sleep-preventing message to the nodes that form its correlated set. After some time, nodes need to check their neighborhood to decide whether to stay asleep or enter working state. Later, a two round and a handshake enhanced versions are proposed as well that achieve smaller working node sets. For the centralized method, a two phase combining first a greedy process to obtain correlation-dominance, and then a Steiner tree process to achieve connectivity, is proposed. A similar conception is used in [77], where the scheduling problem is defined through a redundancy graph, where neighbor nodes are redundant and only one needs to be active. A communication graph completes the problem definition. The authors propose a distributed algorithm that partitions the WSN into smaller regions or *cells* centered around special nodes set as *markers* or *anchors* (much like a hop-distance Voronoi partition of the WSN based on the markers), and prove the proposed solution achieves $(1 + \epsilon)$ optimality. In each cell the scheduling problem can be then solved to optimality (defined as a LP problem, for instance); but since there is no global coordination among different regions, the boundary nodes have suboptimal schedules. This effect is mitigated by using different network partitions by shifting the region boundaries towards the anchor with lower locally-unique identifier value, and obtaining as many schedules. The work is later extended in [76], where the geometric and topologic assumptions are relaxed, and the activity scheduling problem is solved using a similar approach, achieving $1/(1 - \epsilon)$ optimality.

Scheduling can be also combined with other processes of the network; a common issue in this sense is to consider the combined effect of node scheduling together with routing algorithms. In [184] the authors present the Probability-Based Broadcast Forwarding (PBBF) protocol for broadcast optimization in WSNs with active scheduling systems. Two scheduling systems of the literature, IEEE's Power-Save Mode (PSM) and B-MAC, are the models considered for the evaluation of the protocol. PBBF uses two stochastic pro-

cedures: the immediate relay of a received message by a node (with probability p), and the sleep inhibition for a node scheduled to sleep (with probability q). The authors propose an automatic tuning method for p and q, depending on the user's requirement; one can choose two desired properties among the three offered choices: low energy consumption, short latency, and high reliability. The authors of [128] propose the joint optimization of node scheduling and routing for optimal balance between energy consumption and latency. Each link can stochastically fail when a message is being transmitted, and each node can be in sleep mode or not, both resulting in a transmission failure. A centralized algorithm, Semi-Definite Programming-based Convex Polynomial Underestimation (SDPU), and a distributed Simulated Annealing, are used to select the optimal route with respect to an objective function defined as a weighted sum of latency and energy consumed. SDPU is able to find the optimum, but SA is much less computationally expensive.

Finally, a different approach is considered in [97], where the authors consider a rechargeable WSN with quantized energy, in which both the recharge and discharge processes follow a Poisson distribution, with ratio γ between recharge and discharge, and the battery has a limited bucket capacity K. In this problem the lifetime is not the issue, since nodes will eventually recharge some energy, thus they will never stop working completely. Instead, the authors define an *utility* function of the WSN as the probability of an event being detected (which depends on some node having energy at the time of the event); the purpose of scheduling is to maximize the utility value. An aggressive technique that has nodes active whenever they have energy is compared to a threshold technique where a node activates whenever it detects that the number of active nodes has shrunk below a given threshold m. The threshold-based technique is proved to be asymptotically optimal with respect to K when $m = N/\gamma$, where N is the number of nodes in the network.

5.5 Conclusions

In this chapter we have presented the Radio Network Design problem, an NP-hard problem found in the telecommunications field, that is closely related to the domain of WSNs. RND amounts to selecting the locations from a set of available sites, and tuning the configuration parameters of the base stations of a radio system, with the aim of maximizing the radio coverage while reducing the system cost. We have described the most common approaches used to evaluate the coverage, namely the test points and the regular grid models.

We have provided a literature review for the RND problem. There are many works in the literature that use metaheuristics to tackle RND, and most of them use some kind of GA, with multi-objectiveness and parallelism being common issues. Thus, our work related to RND will consist of two parts. In the first we propose a more problem-adapted algorithm, CHC, and its multi-objective version, MOCHC; in the second we propose a self tuning migration method to enhance the performance of a parallel GA. This will be presented in the next chapter.

In the last part of this chapter, we have described the relationship existing between this problem (RND) and a similar problem found in Wireless Sensor Networks, the scheduling problem. This problem consists in selecting different subsets of nodes from a WSN such that each subset offers sufficient service quality (coverage), with the aim of activating the different subsets in a rotational fashion and increase the network lifetime. Finally, we have reviewed the existing literature for this problem (for the sake of completeness), which is considered as one of the most important problems found in the WSNs domain.

Chapter 6

Resolution Methodology and Results for Radio Network Design

In the previous chapter we introduced Radio Network Design (RND) as a problem, reviewed the state-of-the-art for it, and established its link with WSNs. In this chapter we address the approaches adopted to solve the RND problem, and the results obtained.

We start by describing the problem definition of RND that is solved in our work; the definition of RND is not novel, and can be matched to existing definitions in the literature. Both the models assumed for the coverage offered by base stations, the encoding employed for the candidate solutions, and the genetic operators used by the optimization algorithms to explore the search spaces corresponding to RND problem instances are presented and described.

We initially address the resolution of small- to medium-sized problem instances. These instances are designed as first approaches for the problem that serve to test the feasibility of using metaheuristic techniques to solve the problem; in this sense, both mono-objective and multi-objective versions of the problem are taken into consideration. Furthermore, different antenna coverage models are considered, in order to get model-independent results. As a result of this study, we conclude that: (1) metaheuristics are versatile techniques that can tackle the different definitions of the RND problem tested (mono-objective and multi-objective, and using different types of antenna coverage model), and (2) CHC –and its multi-objective counterpart– proved to be highly cost-efficient among the metaheuristics considered.

Then we define a large realistic problem instance, namely the Malaga city RND instance, or Malaga instance for short. Our selected metaheuristic techniques is tested against a wide set of optimization algorithms, within the frame of a joint-work competition, where once again CHC proved to belong in the best performing group.

Finally, we present a novel approach for distributed population-based metaheuristics, based on a theoretical study on the convergence process in a distributed population (see Section 3.4.3), that covers the gap between theory and practice. Our approach consists of a self-tuning technique that automatically and dynamically adjusts the parametric configuration of the migration process in order to achieve a near-optimal balance between exploration and exploitation by seeking population convergence at the end of the execution. This technique is developed for a Genetic Algorithm, since this kind of algorithm was the model adopted for the theoretical study, and is validated through experimental evaluation by using it to solve the Malaga RND instance.

6.1 Problem formulation and models

The definition of the RND problem was already presented in Section 5.1; we briefly review it here. Let us consider a terrain that has to be given coverage; the terrain will be represented by the set L of all potentially covered locations. Then let us consider the set M of all potential transmitter locations. This set is also known as the set of candidate or available location sites, ALS. Let G be the graph $(M \cup L, E)$, where E is a set of edges such that each transmitter location is linked to the locations it covers and let the vector $\vec{x}$ be a solution to the problem where $x_i \in \{0, 1\}$, and $i \in [1, |M|]$. The value x_i is 1 or 0 depending on whether a transmitter is being used or not in the corresponding site. Thus, the *coverage*[1] achieved by the solution $\vec{x}$ of the RND problem can be computed via Equation 6.1, where $M'(\vec{x})$ is the set of transmitter locations selected by $\vec{x}$ from M (we slightly modify the problem formulation in order to include all the potentially covered locations L in the definition, not just those that are actually reachable from the ALS); the total number of transmitters used is simply $|M'(\vec{x})|$ or $|\vec{x}|$.

$$Coverage(\vec{x}) = 100 \cdot \frac{|Neighbors(M'(\vec{x}), E)|}{|L|}. \tag{6.1}$$

In our formulation of RND, the potentially covered locations are taken from a **grid model** discrete representation of the terrain field, for computation of the coverage purposes. Thus, the set L contains regularly distributed locations. The dimension of the grid ($|L|$) depends on the problem instance at hand, and ranges from 287×287 up to 300×450 points. The ALS (or M in the current formulation of the problem) is what truly constitutes the problem instance. The complexity of the problem instance depends on its cardinality (also referred to as size), as the size of the smallest solution space is $2^{|ALS|}$ for the RND problem –which corresponds to a binary choice for each available location. Again, the instance sizes vary throughout this work, ranging from 149 locations, in the smallest instance, to 1000 locations, in the largest one.

The objectives of RND are twofold: the coverage has to be maximized, while the cost, represented by the number of transmitters used, has to be minimized. Besides, the objectives are opposing objectives, since the less transmitters there are, the less coverage they provide. Hence, there are two different approaches considered for this problem:

1. Tackle it as a **mono-objective** problem. This approach requires a scalar function that combines the two goals of the problem; such an objective function $f(\vec{x})$ has been proposed in [31]:

$$f(\vec{x}) = \frac{Coverage(\vec{x})^{\alpha}}{|M'(\vec{x})|}, \tag{6.2}$$

   where the parameter $\alpha > 0$ can be tuned to favor the cover rate item with respect to the number of transmitters. If we set $\alpha = 1$ then the algorithm will not distinguish between a solution with a single antenna producing a coverage C and another with $N >> 1$ antennae producing a coverage $N \times C$. This defeats the purpose of RND since the algorithm would not be searching for solutions that produce high coverages in an efficient way, but only for efficient solutions regardless of the coverage obtained. Therefore, we have to set $\alpha > 1$ in order to guide the search towards solutions with high cover rates. Like Calégari et al. did in [31], we use $\alpha = 2$.

2. Tackle it as a **multi-objective** problem. This is perhaps the most intuitive method to handle opposing objectives. In the multi-objective approach, each of the objectives is set as a separate function for optimization; hence, for RND we have functions $f_1(\vec{x})$ (Equation 6.3) and $f_2(\vec{x})$ (Equation 6.4). The functions have been formulated in order to obtain a general minimization problem. We use the concept of Pareto dominance described in Section 3.5.1 to handle the solutions of a multi-objective problem.

[1] With coverage we are generally referring to relative coverage, that is, the percentage of the terrain that has radio coverage.

$$f_1(\vec{x}) = 100 - Coverage(\vec{x}), \tag{6.3}$$
$$f_2(\vec{x}) = |M'(\vec{x})|. \tag{6.4}$$

Nonetheless, this basic problem definition does not discriminate between solutions as long as one objective is optimized. For instance, a trivial solution with no transmitters is a non-dominated solution since it has zero cost; this is clearly an undesired solution because it is not solving the primary objective of providing radio coverage. Thus, we need to set some *constraints* that will guide the optimization techniques towards solutions that are actually solving the problem; the penalty functions $p_1(\vec{x})$ and $p_2(\vec{x})$, equations 6.5 and 6.6 respectively, represent the minimum required values for each of the objectives, independently of the other objective. These minimum values are $K\%$ of coverage (p_1), and N transmitters at most (p_2).

$$p_1(\vec{x}) = \begin{cases} f_1(\vec{x}) - K & (f_1(\vec{x}) > K) \\ 0 & (f_1(\vec{x}) \leq K) \end{cases}, \tag{6.5}$$
$$p_2(\vec{x}) = \begin{cases} f_2(\vec{x}) - N & (f_2(\vec{x}) > N) \\ 0 & (f_2(\vec{x}) \leq N) \end{cases}. \tag{6.6}$$

The two penalty functions are then combined into a single penalty term $P(\vec{x})$ by simple sum (Equation 6.5).

$$P(\vec{x}) = p_1(\vec{x}) + p_2(\vec{x}). \tag{6.7}$$

The constraint handling works as follows. Whenever two individuals are compared, their constraints are checked. If both are feasible, a Pareto dominance test is directly applied. If one is feasible and the other is infeasible, the former has the lowest rank (lower ranks correspond to "better" solutions). If both individuals are unfeasible, then the one with the lowest amount of constraint violation (Equation 6.7) outranks the other.

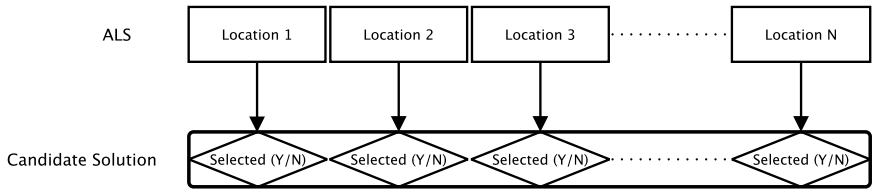
Finally, the three coverage models depicted in Figure 5.4, namely **square coverage**, **circular coverage**, and **sectorial coverage**, have been employed in our formulations of RND. The relationship of the coverage models with the codification required in RND, as well as with the operators employed, are discussed in Section 6.2.

6.2 Representation and operators

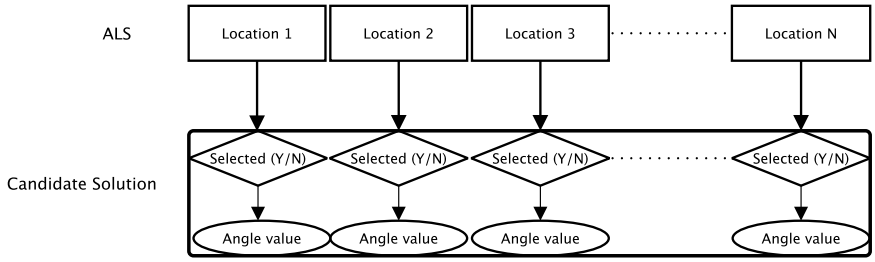
The representation and encoding of the solutions, as well as the genetic operators, depends on the optimization technique, the problem instance, and the model employed for the coverage. We briefly describe them in this section.

6.2.1 Solution encoding

There are two kinds of solution encodings used in RND. They both have a common general structure, of 1 position (or gene) per available location site in the ALS. The first kind corresponds to the two first coverage types (square and circular), which uses parameterless antennae. The second kind corresponds to sectorial antennae, and uses parameterized antennae.



(a) Parameterless



(b) Parameterized

Figure 6.1: Solution encodings in RND: (a) parameterless, (b) parameterized. The ALS is also shown for clarity, though it does not belong to the solution.

Parameterless antennae

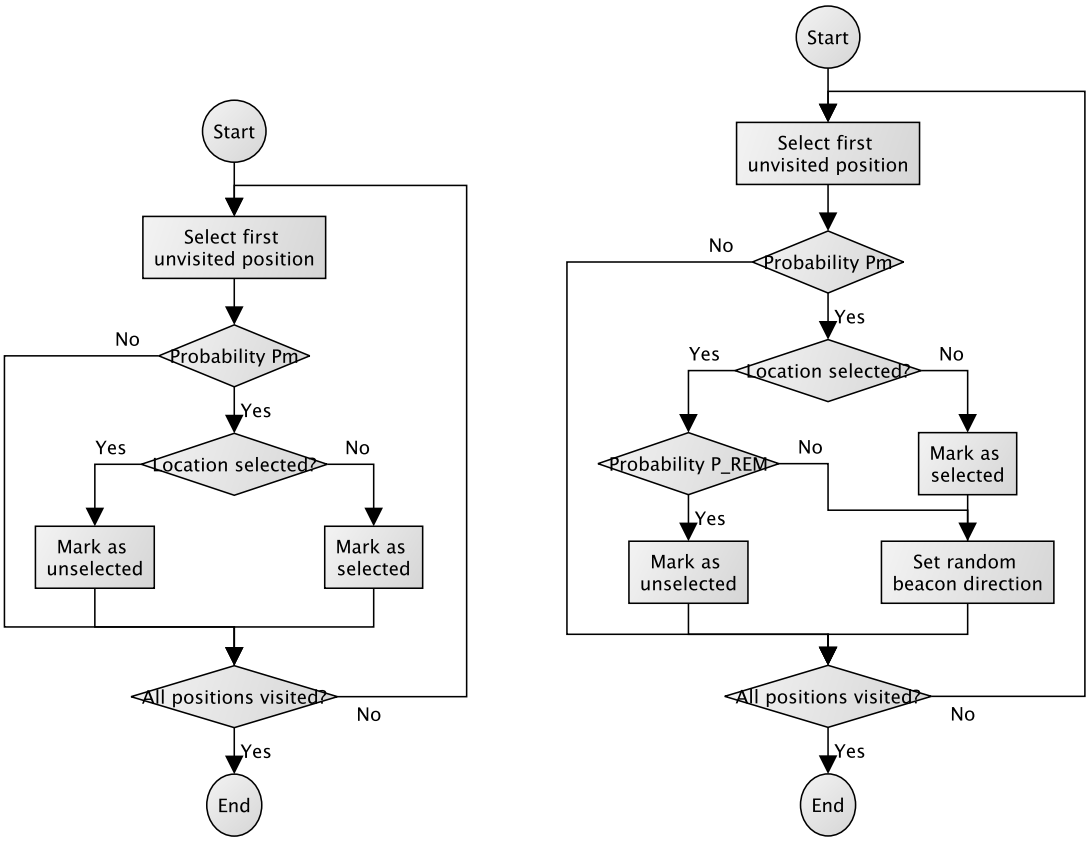
In the first kind of encoding, the only information that needs to be coded into the solution is the set of selected available location sites (from the ALS). This is done with a binary string, where each position of the string corresponds to an available site, and the value of the bit indicates whether the site is selected (1), or not (0). Figure 6.1a illustrates this basic model.

Parameterized antennae

This kind of encoding corresponds to sectorial antennae-using RND instances. In this version of the RND problem, in addition to selecting the set of sites for antenna installation, the *directions* towards which the antennae will point must be selected as well. Thus, a two-level encoding is used; figure 6.1b illustrates this model.

If the sectorial antenna produces a single beam, then the field *angle value* contains a single numerical value in $[0, 360[$; if the sectorial antenna produces three separate beams –the second configuration– then that field contains a vector with three numerical elements in $[0, 360[$. Our formulation of the sectorial coverage uses discretized angular values, with six levels (0, 60, 120, 180, 240 and 300). Note that the angle value only holds a meaning when the site is selected.

Both representations can be unified using a *gene abstraction*. A gene is the amount of information encoded in a given solution with respect to a single available site. Thus, any solution contains as many genes as available sites are contained in the ALS. Then, we can say that for the parameterless antennae problem, a gene is a single bit s, since all the information contained referring to a site is whether it is selected ($s = 1$) or not ($s = 0$); for the parameterized antennae problem, a gene is a tuple $\langle s, \alpha \rangle$, where s is the selection bit, and α contains the values of the antenna angle (we simplify by considering the single beam antenna, extension to three beams is straightforward with three angular values per location site of the ALS).



(a) For parameterless solutions

(b) For parameterized solutions

Figure 6.2: Mutation operators: (a) used with parameterless solutions, (b) used with parameterized solutions.

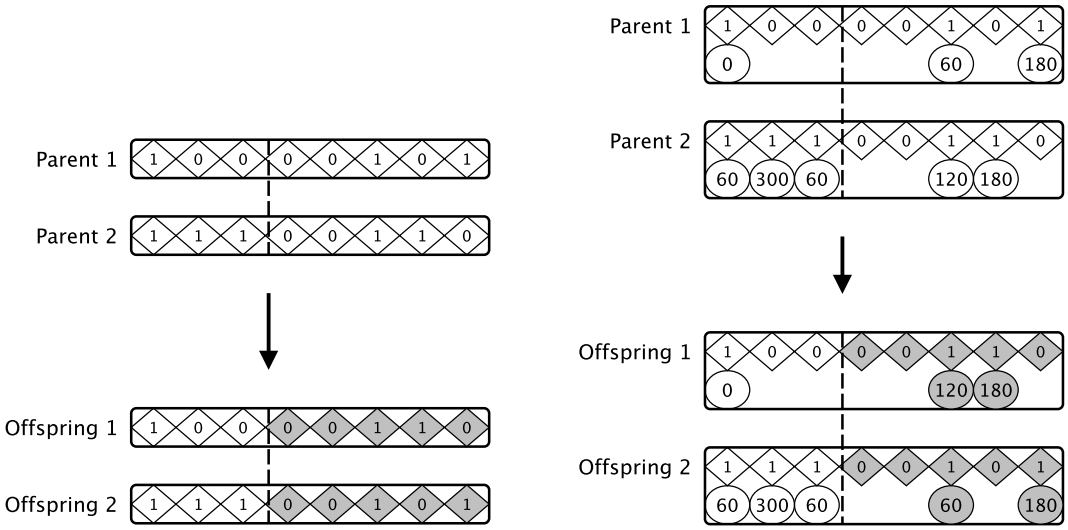
6.2.2 Operators

RND has been solved with SA, GA and CHC algorithms in its mono-objective approach, and NSGA-II and MOCHC in its multi-objective approach. Thus, two kinds of genetic operators are used with RND: mutation operators and crossover operators. The first kind has a single solution input and a single solution output, the second one has a two-solution input and a two-solution output. For each kind of operator there is one version for the parameterless solution encoding and another version for the parameterized encoding. We describe them in the following.

Mutation operators

Mutation operators are employed in GA and NSGA-II as part of the operator pool, in SA as the main method to introduce diversity, and in CHC and MOCHC during the restart mechanism. The same mutation operator is used by all the algorithms.

In the case of parameterless solution encoding, a *bit flip* mutation operator is used. The bit flip mutation visits every position (every bit) of the solution sequentially, and flips it with a given probability p_m (the



(a) Parameterless

(b) Parameterized

Figure 6.3: Single-point crossover examples: (a) parameterless, (b) parameterized.

mutation probability).

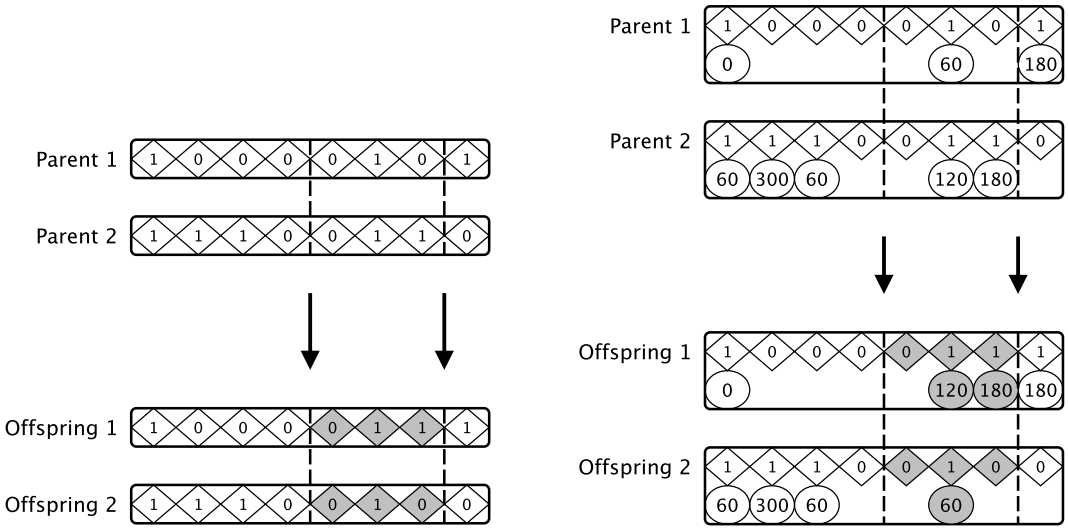
In the case of parameterized solution encoding, a multilevel mutation operator is used. This operator visits every position (bit and angle value) of the solution sequentially, and *modifies* it with probability p_m. The modification procedure in this operator is somewhat more complex than the bit flip of the previous. It works as follows: it the location is not selected, then it is marked selected and a random value(s) is assigned to the angle(s); if it is selected, two things can happen: either it is marked as unselected (with probability p_{REM}), either it is kept selected, but new random value(s) is assigned to the angle(s) (with probability $p_{MOV} = 1 - p_{REM}$). Specifically, we use $p_{REM} = p_{MOV} = 0.5$.

Crossover operators

As said before, a recombination operator, like the crossover, takes two solutions, the parents, and produces two new solutions, the offspring. The general crossover procedure first copies each of the parents into one of the offspring, and then exchanges parts between the offspring. There are several crossover operators employed for RND: single-point crossover, two-point crossover, UX, and HUX crossover. In the following descriptions, the *exchanges* make reference to the parts exchanged between the offspring.

One of the simplest crossover operators is the *Single-Point Crossover* (SPX), used in NSGA-II. This operator works in a similar way for both encodings under the gene abstraction. A point is chosen between two consecutive positions at random in the solution encoding, then all genes beyond the chosen point are exchanged.

The *Two-Point Crossover* operator (TPX), used in the GA, is a more complex version of the preceding operator. This operator selects two positions at random in the solution encoding, and exchanges all the genes comprised between them. Figure 6.4 shows a diagram of the two-point crossover, both for parameterless solutions (Fig. 6.4a) and for parameterized solutions (Fig. 6.4b); the selected positions are marked with



(a) Parameterless

(b) Parameterized

Figure 6.4: Two-point crossover examples: (a) parameterless, (b) parameterized.

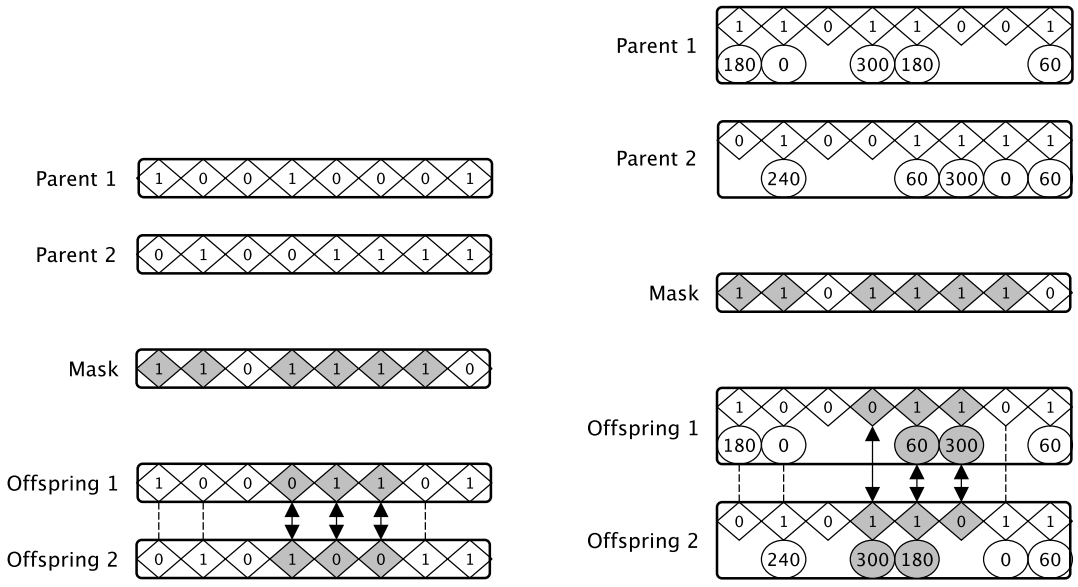
bold dotted vertical lines, and the exchanged parts in the offspring are highlighted with color inversion; in the parameterized solutions, the values of the parameters are shown only for the genes with the selected value to 1, since otherwise the parameter values are meaningless.

The *Uniform Crossover* (UX) is a simple crossover technique in which each gene (which is a bit in the case of parameterless antennae, or the bit with the angle values for parameterized antennae) is swapped between the offspring independently with 50% probability. Thus, each offspring receives on average half the information from each parent. The UX is used in the parallel algorithm, dGA.

Additionally, there is a special crossover operator that was designed for the CHC algorithm ([69]), and is used by MOCHC as well, called *Half-Uniform Crossover* (HUX). In this operator, *exactly* half of the differing genes are swapped: for binary genes, the non-coinciding bits are considered differing genes; for parameterized genes, if either the bit value, or the parametric values when the bit equals 1, are different, then the genes are considered to differ. The exchanged genes are selected randomly among the differing genes. Figure 6.5 shows example applications of the HUX crossover for the two types of solution representations. A bit mask is generated to identify the differing genes between the two parents; each offspring is generated after a parent, then has half of the differing genes (highlighted with gray background) exchanged.

6.3 Problem instances

We classify the RND problem instances into two categories: small-sized "academic" instances, taken or inspired from the literature (we shall call them *test* instances), and a large-sized, real-world inspired instance defined for the city of Malaga (we shall call it the *Malaga* instance).



(a) Parameterless (b) Parameterless

Figure 6.5: HUX crossover examples: (a) parameterless, (b) parameterized.

6.3.1 Test instances

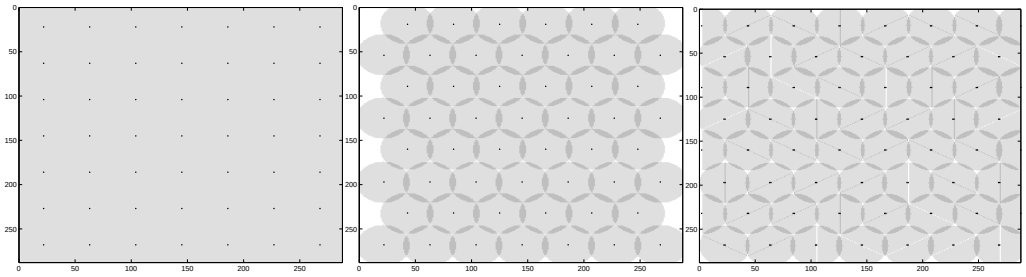
The basic terrain is modeled by a 287×287 grid point model representing an open-air flat area. As said before, the three different antenna types used are: a square shaped cell antenna that covers a 41×41 point cell ([8, 31]), an omnidirectional antenna that covers a 23 point radius circular cell (in order to offer approximately the same coverage as the square antenna), and a directive antenna that covers one sixth of the omnidirectional cell. When directive antennae are employed, three of them are placed in the location site, hence each selected site offers half the coverage with sectorial antennae than with circular antennae.

Five instances of sizes 149, 199, 249, 299 and 349 (corresponding to the number of sites in the ALS) are defined for each type of antenna, except the directional or sectorial one, for which only a size of 149 is used. The generation of the ALS is done as follows:

- For square coverage, 49 sites are placed forming a 7×7 regular grid, and offer full coverage to the terrain; these sites constitute the optimal solution for the mono-objective version, with a fitness value of 204.082, with 100% coverage using 49 transmitters (see Fig. 6.6a). Random locations are added to complete the ALS.

- For circular coverage, 52 sites are placed forming a regular hexagonal grid; these sites constitute the optimal solution to the mono-objective version of RND, with a fitness value of 156.046, with 90.08% coverage using 52 transmitters (see Fig. 6.6b). Random locations are added to complete the ALS.

- For directional coverage, the same locations from circular coverage are used and duplicated (with a small displacement on the copy site), so that the optimal hexagonal grid may be reconstructed with directional antennae (see Fig. 6.6c). Random locations are added to complete the ALS. There are two possibilities for this type of instance: the first, called the *simple* variant, has the restriction that the three sectors must be contiguous, thus forming a single "triple" beam (7 possible configurations



(a) (b) (c)

Figure 6.6: Optimal solutions for the test instances: (a) square coverage, (b) circular coverage, and (c) sectorial coverage.

per site); the second, called the *complex* variant, has absolute freedom for each beam, as long as two beams do not overlap (21 possible configurations per site).

6.3.2 Malaga instance

Real-world radio networks are mostly deployed on urban scenarios to provide coverage for a set a services (GSM, UMTS, etc). An urban scenario has some characteristics that make it different from the basic test instances in Section 6.3.1. In a city, the antennae may only be located in some specific sites, like rooftops or other high places; at the same time there are restricted places like hospitals or schools where antennae may not be placed. A typical urban scenario is non homogeneous and will have regions with more buildings than others, and may also have rivers, parks or some other building-free places.

In this instance, both the terrain and the ALS are generated following the real distribution and geography of the city of Malaga (Spain). The terrain area is modeled by a 300×450 point grid, where each point of the grid represents a surface area of 15×15 square meters. Figure 6.7a illustrates the instance terrain area. The ALS contains $1,000$ sites, corresponding to suitable locations for the installation of radio antennae, as said before. The antenna coverage model is chosen to produce a circular area coverage, with a radius of 30 points (approximately 450 meters, which is a realistic assumption).

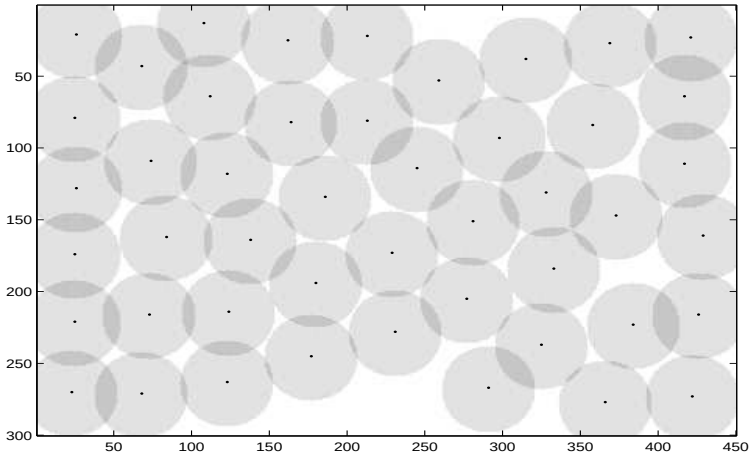
This instance is only solved using the mono-objective approach. The optimal solution is unknown, and full coverage is not achievable.

6.4 Experimental results

In this section we present and discuss the results obtained for the different problem instances tackled in RND. We first present the results obtained in the test instances (described in Section 6.3.1), then describe those obtained for the real-world Malaga instance (described in Section 6.3.2). Table 6.1 sums up the main properties of the instances solved for RND.

As was specified in Section 3.5, every experimental test is executed 30 times, and the results displayed are the average values over the independent runs, unless stated otherwise. A statistical analysis is performed in addition, and its results are shown under a column labeled ‘A’; a ‘+’ sign in that column states that the observed differences are statistically significant.

(a) Aerial view of the city of Malaga



(b) Corresponding coverage of an example solution for the Malaga instance

Figure 6.7: Malaga city instance: (a) map of the area of Malaga, (b) coverage and antennae of the best solution found.

6.4.1 Configuration of the algorithms

We first present the basic parametric configurations selected for the algorithms proposed to solve RND, obtained after an empirical tuning performed over the test instances. These configurations are displayed in Table 6.2.

6.4.2 Test instances

The test instances were solved using two different approaches. First, a mono-objective approach using the fitness function shown in Equation 6.2. Second, a multi-objective approach with objectives shown in equations 6.3 and 6.4, subject to constraints on the coverage and number of transmitters, represented by the penalty functions 6.5, 6.6, combined into a single penalty term (Eq. 6.7).

Table 6.1: Instances solved for RND and their properties.

Type	Antenna	Sites	Search space	Approach	Algorithms
Test	Square	149	$7.136 \cdot 10^{44}$	Mono & Multi	SA, CHC, GA & NSGA-II, MOCHC
		199	$8.035 \cdot 10^{59}$		
		249	$9.046 \cdot 10^{74}$		
		299	$1.019 \cdot 10^{90}$		
		349	$1.147 \cdot 10^{105}$		
	Circular	149	$7.136 \cdot 10^{44}$	Mono & Multi	SA, CHC, GA & NSGA-II, MOCHC
		199	$8.035 \cdot 10^{59}$		
		249	$9.046 \cdot 10^{74}$		
		299	$1.019 \cdot 10^{90}$		
		349	$1.147 \cdot 10^{105}$		
	Sectorial, simple	149	$8.310 \cdot 10^{125}$	Mono & Multi	CHC
	Sectorial, complex	149	$1.025 \cdot 10^{197}$		
Malaga	Circular	$1,000$	$1.072 \cdot 10^{301}$	Mono	SA, CHC

Parameterless antennae

The parameterless antennae instances are those where the coverage model is either square or circular. For these instances, SA, CHC, and GA are used in the mono-objective approach, whereas NSGA-II and MOCHC are used in the multi-objective approach. These instances were found not to be of excessively complexity, and could hence be solved to optimality. For this reason, the stopping criterion was set to finding the optimum, and the comparisons are established among the different algorithms based on their required computational efforts. The results obtained for square and circular coverage antennae with a mono-objective approach are shown in Table 6.3, where the numerical values correspond to the average computational efforts required by the optimization algorithms, which are measured as the number of solutions visited in an execution until the optimum (whose fitness is known beforehand) is found. For each problem instance (instance size and type of coverage model) the best performance is highlighted with gray background.

For the square coverage model, the results are displayed in columns two to five of Table 6.3 (the proposed algorithms are compared with a distributed steady-state GA found in the literature ([7])); column six shows the results of the statistical analysis. CHC is the algorithm that requires the lowest computational effort to solve the RND problem to optimality for the five instance sizes defined, followed by SA. GA and dssGA are far behind, requiring computational efforts that are at least one order of magnitude higher than that of CHC. All the differences were found to be statistically significant.

For the circular coverage model, the results of the proposed algorithms are shown in columns seven to nine of Table 6.3, with the statistical analysis results in column ten. Again CHC requires the smallest effort to solve the problem to optimality, followed by SA, and GA is far behind, all differences being statistically significant. These instances are more difficult to solve by the optimization techniques, as can be noticed by the increased values of computational efforts in all the algorithms: CHC requires between a 30% (for size 149) and a 239% (for size 349) additional effort to solve these instances.

The multi-objective approach to RND was solved using the multi-objective version of CHC, MOCHC, and the state-of-the-art technique in the multi-objective domain field, NSGA-II. In order to maintain the consistency with the results shown previously, the executions were run until the optimal solution of the equivalent mono-objective problem was found. The results for the different instance sizes using square or circular coverage models are displayed in Table 6.4, where the numerical values correspond to the average computational efforts required by the optimization algorithms, which are measured as the number of solutions visited in an execution until the optimum (whose fitness is known beforehand) is found. For each problem instance (instance size and type of coverage model) the best performance is highlighted with grey

Table 6.2: Parametric configuration of the optimization algorithms used in RND.

Algorithm	**GA**
population	100
selection	roulette
crossover	$\begin{cases} TPX \\ p_c = 0.80 \end{cases}$
mutation	$p_m = 1/L$
replacement	elitist

Algorithm	**NSGA-II**
population	100
selection	roulette
crossover	$\begin{cases} SPX \\ p_c = 0.80 \end{cases}$
mutation	$p_m = 1/L$
replacement	ranking and crowding

Algorithm	**CHC**
population	100
selection	$\begin{cases} \text{incest prevention} \\ \text{threshold} = 25\% \end{cases}$
crossover	$\begin{cases} HUX \\ p_c = 0.80 \end{cases}$
replacement	elitist
restart mutation	$p_m = 0.35$

Algorithm	**MOCHC**
population	100
selection	$\begin{cases} \text{incest prevention} \\ \text{threshold} = 25\% \end{cases}$
crossover	$\begin{cases} HUX \\ p_c = 0.80 \end{cases}$
replacement	ranking and crowding
restart mutation	$p_m = 0.35$

Algorithm	**SA**
mutation	$p_m = 1/L$
Markov chain	50
cooling	$\alpha = 0.99995$

Table 6.3: Computational effort of the mono-objective techniques (number of evaluations).

Instance	**Square coverage**					**Circular coverage**			
size	**SA**	**GA**	**CHC**	**dssGA8** [7]	A	**SA**	**GA**	**CHC**	A
149	8.676e+04	1.419e+05	1.335e+04	7.859e+05	+	8.318e+04	2.066e+05	1.736e+04	+
199	1.970e+05	4.105e+05	2.465e+04	1.467e+06	+	2.623e+05	1.152e+06	4.696e+04	+
249	3.341e+05	9.871e+05	3.903e+04	2.481e+06	+	9.136e+05	3.354e+06	8.577e+04	+
299	6.380e+05	1.892e+06	5.408e+04	2.998e+06	+	2.946e+06	8.081e+06	1.512e+05	+
349	8.108e+05	3.612e+06	7.022e+04	4.710e+06	+	6.136e+06	1.999e+07	2.377e+05	+

background.

Once more the best results were obtained with the multi-objective version of CHC, MOCHC. When square coverage model is used, the computational effort required by MOCHC is half that of NSGA-II, while for circular coverage model it is almost an order of magnitude less for MOCHC than for NSGA-II; all differences are statistically significant.

Thus, CHC has been found to be the best performing algorithm in the mono-objective and multi-objective approaches for this problem, for both square and circular coverage models. In the mono-objective approach, the computational effort required by CHC to solve the different instances of the problem is almost an order of magnitude lower than that of the rest of techniques, all the observed differences were found to be statistically significant. GA offered the poorest results, requiring enormous amounts of computational effort to solve the RND problem, while SA showed an intermediate behavior. In the multi-objective approach, the computational effort required by MOCHC is half that of NSGA-II for square coverage, and an order of magnitude lower than that of NSGA-II for circular coverage. Thus, CHC/MOCHC is chosen as the best performing algorithm for the RND problem.

Table 6.4: Computational effort of the multi-objective techniques (number of evaluations).

Instance size	Square coverage			Circular coverage		
	MOCHC	NSGA-II	A	MOCHC	NSGA-II	A
149	1.814e+4	3.745e+4	+	2.8272e+04	1.81508e+05	+
199	3.998e+4	7.479e+4	+	7.7773e+04	8.16206e+05	+
249	7.723e+4	1.418e+5	+	2.6227e+05	1.747044e+06	+
299	1.136e+5	1.987e+5	+	5.6581e+05	2.665930e+06	+
349	1.574e+5	2.871e+5	+	9.0558e+05	4.137630e+06	+

Directive antennae

When using directive antennae, the exact optimal solutions are unknown *a priori*, hence executions are now run until a predefined number of solution evaluations is met, and the obtained fitness values are averaged over the total number of independent executions. This differs from the previous experiments, where the executions were run until the optimum was found. The chosen value for the number of evaluations is $1,000,000$.

There is a second variation regarding the fitness function: the number of transmitters used is replaced with the number of location sites selected (there are three transmitters per location). Though this does not affect the search behavior (it escalates the fitness value by a constant factor of three), it makes the results to be more intuitive for comparison purposes since the fitness values are more closely related to those in the previous experiments. In theory, sectorial cells have half the efficiency of circular cells (they cover exactly a half-circle), so an equivalent solution should produce half the fitness value (requiring double the number of sites to obtain the same coverage). Given that the optimal fitness was 164.672 for omnidirectional antennae, we should expect an optimum solution for this problem to produce a fitness value of approximately 82.336.

Table 6.5: Results of the study for CHC using directive transmitters.

Problem	Simple Version		Complex Version	
	Mono-objective	Multi-objective	Mono-objective	Multi-objective
Best fitness	85.328	85.750	80.693	84.766
Average fitness	84.884	84.613	78.787	82.616
Worst fitness	84.628	83.164	76.211	78.627

Table 6.5 shows the results obtained for the RND using directive antennae (the two cases), in terms of fitness values produced. For the multi-objective approach, the fitness value of each solution of the obtained set is calculated, and the highest value is kept for each execution.

We first remark that the equivalent fitness value of the circular coverage model optimum, 82.336, is outperformed in the majority of scenarios (in all the scenarios for the simple version of the problem). This is due to the fact that, besides reconstructing the hexagonal grid, the algorithm can efficiently improve that structure by adding sites at the border of the terrain to cover the holes in the frontier (see figures 6.6b and 6.6c).

The second remark is that, although for the simple version of the problem both the mono- and the multi-objective approaches produce solutions of similar quality (with no significant differences between fitness values), for the complex version the multi-objective approach produces solutions with significantly higher fitness values. It seems thus that for higher-dimension problems, the multi-objective approach offers a better exploration of the search space.

6.4.3 Malaga instance

The Malaga instance was solved using SA and CHC, which were compared against a wide set of state-of-the-art optimization techniques proposed by three other collaborating research groups, including: Iterated Local Search (ILS), Population-Based Incremental Learning (PBIL), Clustered Genetic Algorithm (AGC), Clustered Chromosome Appearance Probability Matrix GA (CAPMC), Clustered Memetic Algorithm (MAC), Differential Evolution (DE), Greedy Randomized Adaptive Search Procedure (GRASP), Variable Neighborhood Search (VNS), and hybrid and multi-start variants of the techniques (MS_FNS, HYBRID_RUFNS, GRASP_EVNS, MS_GEPVNS, GRASP_SRCL). The results used for comparisons correspond to executions run until $5,000,000$ solutions were evaluated.

Figure 6.8 shows the average values and standard deviations of the fitness values obtained by the 14 algorithms used, sort from left to right in descending order of performance. As it can be seen, CHC ranks third in performance, while SA ranks on the twelfth place. Figures 6.9a and 6.9b plot the average execution traces of the upper and lower quartile algorithms, to which CHC and SA belong, respectively.

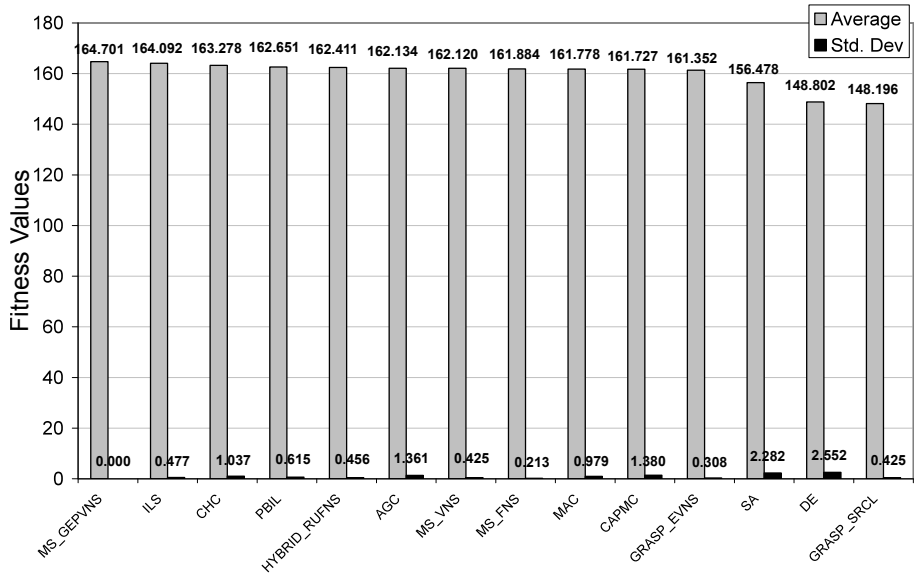


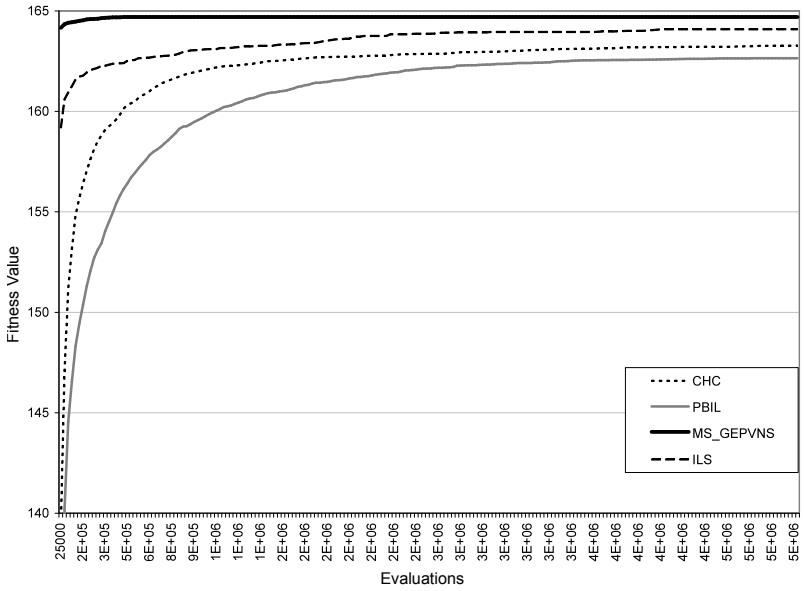
Figure 6.8: Results obtained for the Malaga instance.

CHC proves thus to be a very high-performing technique for the RND problem. Though it was not the best performing technique of all the 14 algorithms tried, it came up third and close to the best ones; additionally, it has to be stated that the CHC algorithm was in its canonical form, without specific problem-related operators or enhancements, unlike other techniques of the pool[2]. Hence we believe there is still room for improvement in CHC, which is a highly promising technique for RND.

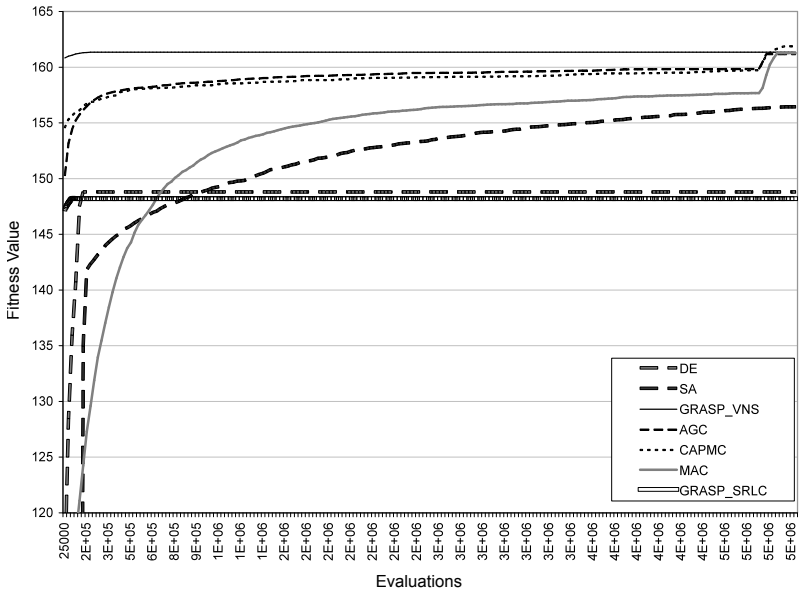
6.5 Self-adaptive distributed technique for RND

We present in this section a novel contribution that is aimed at improving the conditions of use for distributed optimization algorithms, and validate it by applying it to the RND problem, under its most complex instance: the Malaga instance.

[2]MS_GEPVNS, for instance, takes advantage of the problem definition and allows for partial reevaluation of a solution to evaluate its neighbors, thus effectively visiting *more* solutions (though within a close neighborhood) without taking it into account for the computational effort



(a) Upper Quartile



(b) Lower Quartile

Figure 6.9: (a) Upper and (b) Lower quartile execution traces obtained for the Malaga instance.

Our contribution consists in an automatic mechanism to configure the migration schedule of a distributed algorithm, such that the migration periods are set and updated *on-the-fly* during execution, and the resulting algorithm is robust and high-performing. The mechanism is based on a theoretical study of the effect of migrations over the different subpopulations of a distributed genetic algorithm, especially focused on the convergence properties (see Section 3.4.3). We extrapolate the results of this analytic study to generate a theory-driven designing technique. Our contribution can help reduce enormously the tuning times for this kind of technique by removing the migration-related parameters from this process; this can be a decisive factor when using distributed techniques in the presence of hard time constraints.

For the validation, a distributed GA, or dGA (see Section 4.1.3), is used as the recipient to test the automatic migration tuning technique. The topology employed is a directed ring topology. Each element of the ring, i.e., each subpopulation is also called an *island*; there are eight such islands, each of which will host a population of 50 individuals. The corresponding sequential/panmictic population contains 400 individuals. As in Section 6.4.3, the executions are run until $5,000,000$ solutions are evaluated, the number of generations is thus $12,500$. In the migration process every island sends its best individual, and includes the received individual in its subpopulation replacing the worst individual (elitist criterion). The basic configuration parameters of the dGA are shown in Table 6.6.

Table 6.6: Set of Configuration Parameters for the Sequential Genetic Algorithm

Parameter	**Value**
population size	50 or 400
mutation	bit-flip, $p_m = 1/L$
crossover	uniform, $p_c = 0.60$
selection	*random* or *roulette*
replacement	*tournament* or *elitist*

We define two configurations of (d)GA, regarding the selection and replacement operators used, in order to get different combinations of intensity and diversity in the search process. The first one emphasizes diversity, and combines a random selection of the parents and a four-tournament selection of the next generation. This configuration shall be referred to as the "Normal" selection hereafter. The second one has a special stress on intensity -if is therefore a much more elitist selection-, it combines roulette selection for the parents, and elitist selection of the next generation. This configuration shall be referred to as "Elitist" selection hereafter.

We first describe our proposed technique for automatic adaptation of the migration schedule in Section 6.5.1. Then, we present the results obtained by the technique, and compare them against the results obtained with equivalent sequential GAs and parallel dGAs running with fixed migration schedules in Section 6.5.2.

6.5.1 Application of the model

As stated before, the basis for our automatic migration-tuning technique is the model described in Section 3.4.3. More specifically, the model provides a tool for automatically tuning the migration schedule of our dGA. The key idea is that setting the migration schedule in such a way that the algorithm converges towards the end of the execution should improve the algorithm's performance. The rationale behind this is to balance diversity and intensity: premature convergence results in excessive intensity (local optima), while lack of convergence results in excessive diversity (slow search).

According to the theoretical model developed in [11], given an initial population and a regular migration schedule, the rate of convergence of the global population is given by Equation 3.23, and the takeover time

by Equation 3.24. In the migration-tuning technique, the process has to be undergone in the opposite direction: given a target takeover time (we wish that takeover happens upon execution completion, $t^* = t_{execution}$), how to set the migration schedule in order to achieve it?

Assume that during an execution, full convergence is attained at the end (takeover happens at the end of the execution). Furthermore, assume that every island containing the optimum converges before the next migration (as is the case in Figure 3.11). At some given time $t_{current}$ during that execution the percentage of the population conquered by the optimum is $P(t)$. Then, according to the model, we have:

$$t_{execution} = per \cdot d(T) - \frac{1}{b} ln \left(\frac{1}{a} \frac{\epsilon}{N - d(T) - \epsilon N} \right), \tag{6.8}$$

at any time, the "progress" of the execution is $\frac{P(t)}{1/N}$, the equivalent execution time is $per \cdot \frac{P(t)}{1/N}$, from which we get:

$$t_{remaining} + per \cdot \left(\frac{P(t)}{1/N} \right) = per \cdot d(T) - \frac{1}{b} ln \left(\frac{1}{a} \frac{\epsilon}{N - d(T) - \epsilon N} \right). \tag{6.9}$$

From this, the value of the migration period per can be extracted as:

$$per = \frac{t_{remaining} - K}{d(T) - \left(\frac{P(T)}{1/N} \right)}, \tag{6.10}$$

where we have defined

$$K = \frac{1}{b} \cdot ln \left(\frac{1}{a} \cdot \frac{\epsilon}{N - d(T) - \epsilon N} \right), \tag{6.11}$$

where a is set equal to the size of a subpopulation hosted by an island, $b = 0.4$, and ϵ is the tolerance parameter that we set as $\epsilon = 0.1$.

However, the model makes two assumptions that cannot be met in real scenarios where GA is to be applied. The first one is that the optimum is *already present* in the initial population. The second assumption is that *only selection operators* are employed (thus no mutation and no crossover). This means that solutions do not evolve throughout the algorithm execution and no new solutions can be produced. Thus, the theoretical model requires an extension in order to cover the distance from theoretical conditions to the conditions in a practical scenario.

We propose three extensions of the base model to overcome the limits of the theoretical assumptions:

1. First, in a real scenario the optimal fitness value is unknown. Therefore, a **target or objective fitness** value has to be determined beforehand, i.e,. a fitness value high enough so that any solution producing that fitness (or better) can safely be considered as *fit*. In a real problem, this task can be completed by defining the minimal requirements in the solution. In the present case, that parameter is set to a large value close to the the largest known fitness value. This target fitness acts as a threshold value, meaning that any solution producing a fitness value over it will be considered as an optimum whenever the takeover or the growth are checked.

2. Second, the optimal solution is not present in the initial population (and may *never* be in the population). Instead, the method will search for the **best solution present in the population**. The ratio between that solution's fitness and the objective fitness value is directly applied to the period value obtained from Equation 6.10 as a multiplicative corrective factor.

3. The third extension is an attempt to deal with the unpredictable nature of the crossover and mutation operators, and the fact that it is impossible to know in advance the kind of new solutions that will appear through the search process. Due to this, the tuning process will try to force the convergence process so that the optimum can appear in one island, and have sufficient time left to propagate to the rest of the islands; for this, it reduces the migration period by a **factor equal to the total number of islands**.

The resulting modified value of migration period $per^*$of the automatically tuned dGA is shown in Equation 6.12.

$$per^* = \frac{fitness_{best\ found}}{fitness_{objective}} \cdot \frac{per}{N}, \tag{6.12}$$

where $fitness_{best\ found}$ is the best fitness found in the global population, $fitness_{objective}$ is the predefined target fitness (that is considered to be *optimal*), and N is the number of islands.

In our technique, a common migration schedule is set for the whole distributed population, but for its calculation some global knowledge about the whole population is required. Hence, in the practical implementation of the method a master process gathers all the relevant information from the islands, makes the calculations, and then sends the results (i.e., the new migration period) back to all the islands. The information gathered by the master process includes all the fitness values of the individuals present at the subpopulations. Since the technique controls the migration parameters, the whole process will take place once after each migration: every island will first perform the migration process (send and receive the migrating individuals), then send all its fitness values to the master process, then block itself. When the master has gathered the information from all of the islands, it calculates the new migration period, and sends it back to every island. When the islands receive this information from the master, they update their configuration with the new migration period, then resume their execution.

6.5.2 Results of the proposed technique

This Section discusses the results produced by the automatically tuned migration period on a dGA. The two selection methods described in the beginning of the section are used. The results obtained are shown against the best results produced by the sequential executions, and the distributed executions with constant migration periods.

Comparison with sequential GAs

We have defined two types of sequential GA. In the first one, the algorithm handles a pool of solutions equal to the global pool of solutions of the distributed algorithm. In the second one, the pool will be equaled to a single island's pool of solutions of the distributed GA. In order for the comparisons to be fair and meaningful, all the executions are sized after the total number of single solution evaluations (which we set to $5,000,000$) instead of the number of iterations. Thus, the two sequential experiments described above handle a pool of 400 solutions and perform $12,500$ iterations in the first case, and a pool of 50 solutions and perform $100,000$ iterations in the second. In total, there are four different configurations for GA: two selection mechanisms, with two population sizes each.

The average traces of the sequential executions are shown in Figure 6.10, where they are compared to the adaptive technique. The techniques are labeled with 'N' if they use the Normal selection method, and 'E' if they used the Elitist selection. The number in the labels of the sequential algorithms make reference to their population size.

Several observations can be made about the obtained results regarding the sequential GAs. First, surprisingly, the size of the population does not have a significant effect on the quality of the solutions produced by the GA. In fact, when Normal selection is used a population size of 50 individuals produces better results than 400 individuals, whereas when Elitist selection is used the opposite is true. For each of these two selection methods the differences between the results are small. Second, the Elitist selection procedure outperforms the Normal selection procedure. Any configuration with Elitist selection obtains higher fitness values than both configurations with the Normal selection. The lowest margin is between Elitist with 50 individuals (fitness of 157.001) and Normal with 50 individuals (fitness of 156.249). The largest margin happens between Elitist with 400 individuals (fitness of 157.95) and Normal with 400 individuals (fitness of 155.763).

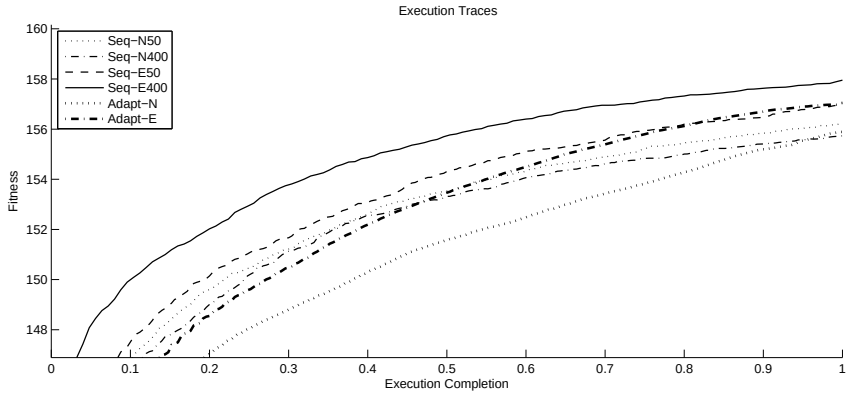


Figure 6.10: Average execution traces of the adaptive migration technique compared with the sequential executions of GA.

As a secondary conclusion, the benefits given by a balance between intensity and diversity are illustrated by these results. Since the Normal selection procedure is biased towards diversity, a small population size suits it best, as small populations tend towards fast convergence hence intensity; at the same time the Elitist selection is biased towards intensity, therefore larger population sizes favoring diversity provide the best combination.

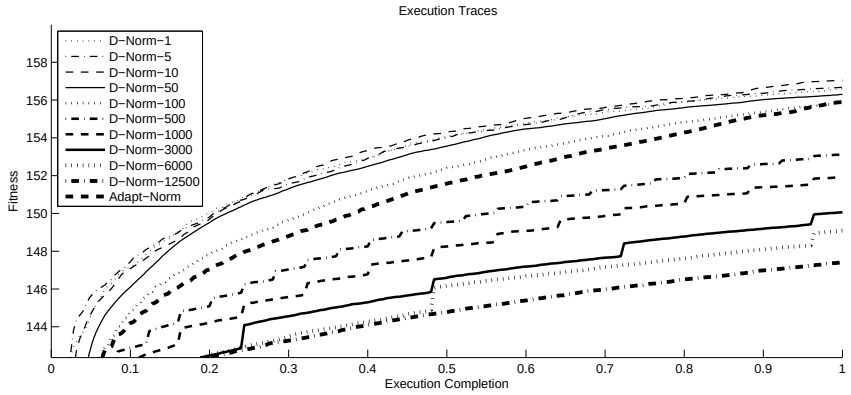
Regarding our proposed adaptive technique, we note that both configurations, the one using Normal selection and the one using Elitist selection, show a similar behavior: they start with low fitness values, but they increase the fitness value more consistently towards the end of the execution than the sequential GAs; the Elitist selection produces consistently higher fitness values than the Normal selection, and the difference seems to be stable over the execution. In fact, the adaptive technique manages to outperform all sequential configurations except for the one using a population of 400 individuals and Elitist selection.

Comparison with dGAs with fixed migration schedules

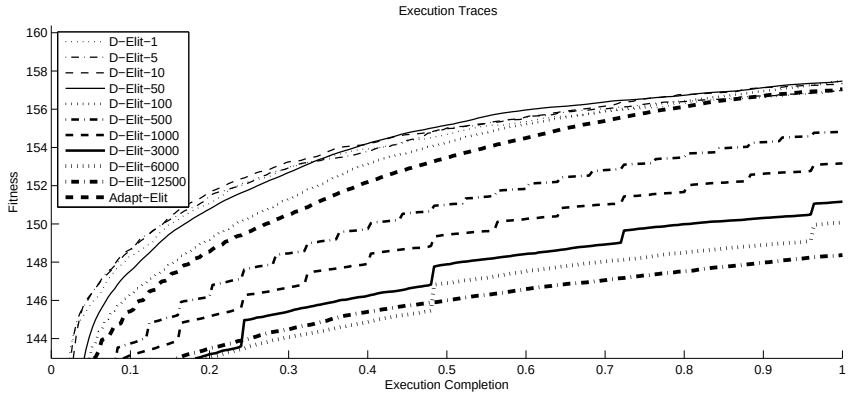
In dGAs with fixed migration schedules, the migrations take place regularly once every k iterations, where k is the migration period. We use the two selection mechanisms described above in these experiments, working within the local GA running in every island. A total set of 10 different values is employed for the fixed migration period, ranging from 1 (constant communication among islands) to $12,500$ (complete isolation); this is done in order to determine the relative effectiveness of different fixed migration schedules, and to compare these schedules with our proposed self-adaptive schedule. In total, there are 20 different configurations for dGA with fixed migration schedules: two selection mechanisms with 10 migration periods each.

The results for each selection mechanism are first discussed separately, then compared. The execution traces obtained with the Normal selection procedure are displayed in Figure 6.11a, those obtained with the Elitist selection procedure are shown in Figure 6.11b. In both cases, the corresponding adaptive migration technique is displayed as well, labeled 'Adapt', whereas the configurations with fixed migration schedules are labeled 'D'; in turn, the Normal selection method is labeled 'Norm', while the Elitist selection is labeled 'Elit'; finally, the numbers in the labels indicate the value of the migration period used (for the 'D' configurations using a fixed period).

When the Normal selection method is employed, the best results are obtained using low migration periods. The final average fitness values produced by the dGA with migration periods ranging from 1 to 100 are quite similar (all above 156), and for higher periods it gradually deteriorates (all below 154),



(a)



(b)

Figure 6.11: Comparison of the adaptive migration technique with parallel executions of dGA.

reaching its lowest for a period of $12,500$ (fitness of 150.60). The migration period of 10 produces the best fitness, closely followed by 5 and 1, with respective fitness values of 157.098, 156.707 and 156.631; the relationship among the best performing configurations seems to be quite stable throughout the execution.

When the Elitist selection is used, the best results correspond to migration period values of 100 (first) and 50 (second), with average fitness values of 157.631 and 157.540, respectively. In general, these results are better than those of the Normal selection method. By comparing the behavior of these two configurations during their execution time it can be observed that, unlike the case with Normal selection, here the migration period of 50 produces higher average fitness than a period of 100 halfway into the execution, but at the end this trend reverses. As a derived effect, should the length of the execution be extended, the configuration using a migration period of 100 is expected to produce increasingly better results than the one using 50. Therefore, if the execution length had to be maintained at low values (below $5,000,000$ solution evaluations), a migration period of 50 should be selected; if the execution length is high ($5,000,000$ evaluations or beyond), a migration period of 100 offers better performance.

Regarding the adaptive method, a similar effect as the one observed previously can be noticed: at first, the high-performing fixed configurations produce clearly higher fitness values than the adaptive, but towards the end the adaptive technique experiences a faster growth and is able to catch up with the former. In the case of the Normal selection, the adaptive technique is barely capable of reaching the high values of fitness, but

with Elitist selection it has clearly got to a similar level of performance, and is only slightly outperformed by fixed migration values of 50 and 100 at the end.

Analysis of the migration period

We focus in this section on the behavior of the adaptive technique when the Elitist selection method is used, since that combination is the one that seemingly produces the best results. Figure 6.12 shows the values adopted by the migration period during the execution time using the automatic tuning technique. The best found fixed values (50 and 100) are also represented as horizontal dashed lines as a reference. As can be seen in the plot, the migration period starts at a low value (somewhere around 27 iterations) and suddenly rises up to 200 iterations; then it gradually decreases during the execution until it reaches a preimposed minimum -equal to the number of islands- by $12,000$ iterations (thus, after 95% of the execution is complete). Single execution values are plotted in an overlap fashion instead of an averaged value for two reasons: first, since all the executions display a similar behavior, there is no added confusion by the overlap, second, an averaged representation might be misleading, as there are two single executions that display an "anomalous" behavior towards the end of the execution by suddenly rising up to values of approximately 300 and 150; this happens when "optimal" solutions are found before the expected time (which is the end of the execution), and the algorithm can thus settle with a lower migration frequency (higher period) since the takeover is likely to happen. We remark that the overall values of migration period are comprised within a reasonable range of values (between 8 and 200) around the best known values (50 and 100), which proves the proposal's correctness.

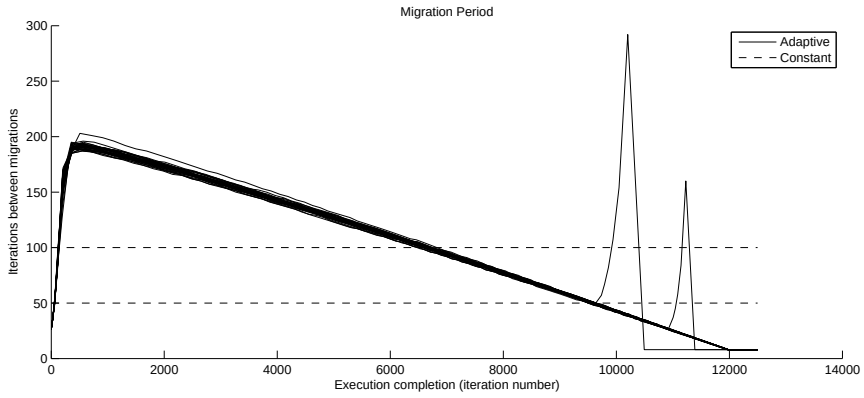


Figure 6.12: Values given to the migration period by the automatic tuning method on dGA with Elitist selection.

The explanation for this behavior is as follows. At the beginning of the execution, the dominant factor is the fitness ratio $\frac{fitness_{best\ found}}{fitness_{objective}}$ (the best fitness values in the initial population are around 17, as opposite to the objective fitness which is set to 160). Then the fitness value of the best solution quickly increases producing the observed rise in the migration period (the average fitness is 120 after $100,000$ evaluations, or 250 iterations). Then, as the number of remaining iterations reduces, the migration period progressively diminishes as observed in Figure 6.12, until it reaches the lower bound.

Analysis of the computation times

This section quantifies the computation time ratios among the different algorithms used. Three sets of (time) data are compared, all of which were obtained on the same machines (in order for the comparison to

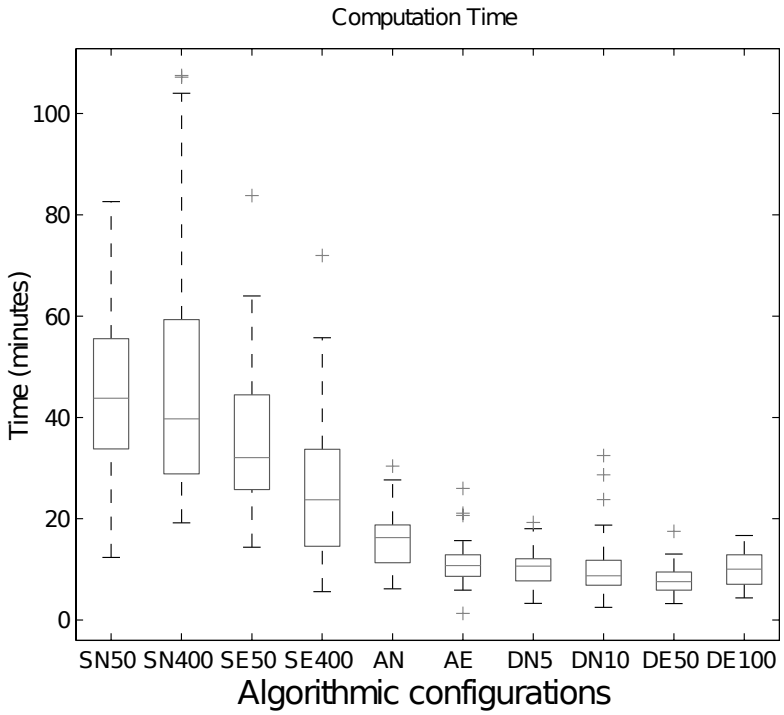


Figure 6.13: Computational times required to reach the fitness threshold for the best configurations found. 'S' stands for Sequential, 'A' is Adaptive, 'D' is Distributed with fixed migration schedule. 'N' is Normal selection and 'E' is Elitist selection.

be meaningful). The first set includes the four sequential executions. The second set includes the twenty distributed executions with constant migration schedules running on four identical machines -hosting two islands per machine. The third set includes the two distributed executions with adaptive migration schedule running of four machines.

Since the executions were stopped after a predefined number of solution evaluations $(5,000,000)$, a straight time comparison among the different configurations is not very significant, and can be misleading (since the quality of the solutions obtained is not the same every time). Therefore a threshold fitness value is set, the time required by the different algorithmic configurations to reach that threshold are checked. To be more specific, the threshold has to be high enough to ensure that the corresponding solution is acceptable, and low enough to ensure that all the executions taken into account for comparison are able to reach it. According the these criteria, the selected value is 151.9, equal to 92.2% of the highest fitness known.

A wide comparison of the computation times is offered in Figure 6.13, where the results obtained by the sequential and distributed configurations are both included. In the labels, 'S' stands for sequential, 'D' for distributed, and 'A' for adaptive migration; ''N' and 'E' stand for Normal and Elitist selection mechanisms, respectively; finally, the numbers indicate the population size in the case of sequential algorithms, or migration period in the case of distributed ones.

As expected, the times required by the sequential configurations of GA are significantly higher than the ones by the distributed configurations. Among the sequential configurations, the one with Elitist selection and a population of 400 individuals (fourth box) has clearly better time response than the other three, though not as good as the distributed ones. Since that configuration was also the one that obtained the best results it seems obvious that this configuration has to be selected should a sequential GA be used to solve this

problem.

Regarding the distributed configurations, the adaptive schedule using Normal selection contains sensibly higher times than the rest of the boxes, thus revealing a poorer performance. The best performance is displayed by the constant migration every 50 iterations and Elitist selection, though no significant differences are found among the five best-performing configurations. This should come as no surprise since, although the best results were obtained with the Elitist selection and a migration period of 100 iterations, the trace plot in Figure 6.11b shows that only at the end of the execution does that configuration outperform the one with a period of 50 iterations.

As a result of this section, it can be stated that the use of this work's proposal (parallel distributed GA with adaptive migration schedule) doesn't show significant drawbacks in running wall clock time over the best found configurations for distributed GA with constant migration schedules. Moreover, as expected, the execution time of the distributed approaches outperformed amply those of the sequential approaches. Finally, although the results produced by the adaptive migration period do not outperform the best results achievable using a fixed schedule, the cost of finding such a schedule compensates this effect, as is sketched in the following section.

Advantages in terms of parameter tuning cost

The last discussion is centered on the effective alleviation of the parameter tuning cost that the automatic tuning technique provides. For that, an alternative scenario is considered for comparison. In this scenario, a network designer wishes to tune the dGA migration period, and for this purpose will perform a (binary) search process. In this process, whenever a range of possible values is considered for the tunable parameter, the two extremes and the middle value are evaluated. Then, the best half-range (out of the two subranges defined by the middle value) is kept. The process is repeated until a value is found that produces comparable results to the proposed technique, at which time the process is stopped. This study will be performed only for the configuration using Elitist selection, since it is the one that has produced the best results.

Each tested value requires 30 independent executions to be performed for statistical confidence. The adaptive technique requires approximately 15 hours of wall clock computation time to perform 30 independent executions (counting 30 minutes per execution).

For dGA with Elitist selection, the range of migration period values where the constant migration outperforms the adaptive one is $[10 - 100]$. This range is reached after 9 steps using a *linear* search (the next value tested is the arithmetic mean of the two values defining the range), constituting a total number of 270 independent executions (nine times the total number of executions performed with the adaptive migration schedule). In wall clock time that equals approximately 135 hours of computation using four computers like the ones used in this work.

If a *logarithmic* search strategy is adopted (the next value tested is not the arithmetic mean value of the extreme values of the selected range, but the square root of the product of these values instead), then the "right" configuration can be attained after only 4 steps, or 120 independent executions. This search strategy is thus more efficient, but still requires 60 hours of computation nonetheless.

In summary, with the automatic adaptive technique the results from the 30 independent executions can be obtained after 15 hours of computation. If the migration schedule is empirically tuned (which is the case in most existing similar work), equivalently good results can be obtained after 135 hours of computation if linear search is used, or after 60 hours in the case of the logarithmic search.

Therefore, our proposed automatic tuning techniques produces savings of between 75% and 89% of the overall computation time when compared to a search method requiring migration parameter tuning (in this case, tuning of the migration period parameter).

6.6 Conclusions

In this chapter we have formulated and solved the RND problem, which consists in selecting the location sites for the installation of radio transceivers in order to provide coverage to a given terrain area. We have presented several problem instances, ranging from academic test instances using various antenna coverage models to a large real-world based instance, the Malaga instance.

A set of medium-sized test instances are used as the testbench to assess the capability of different metaheuristics to tackle this kind of problem. In this sense, they are solved under the mono-objective approach using SA, CHC and GA, and under the multi-objective approach using NSGA-II and MOCHC, the multi-objective version of the CHC algorithm that was specially developed to solve this problem. The results in both fields for the multiple instance types (with different antenna models) and sizes defined highlight CHC and MOCHC as high-performing techniques, that achieve very good results even when applied under their canonical form to different variations of the problem at hand.

These initial results are later extended in the work performed for the Malaga instance, where two of our techniques, SA and CHC, are tested against 12 different advanced optimization algorithms proposed by three other collaborating research groups, which include hybridizations and multi-start variants, applied to a RND problem instance of high dimension. The results demonstrate that CHC is competitive even compared against specially tailored techniques with notable enhancements such as partial solution (re)evaluation; as a matter of fact, CHC ranked in the upper quartile.

Finally, following a novel philosophy, we propose a theory-to-practice contribution in the domain of parallel optimization algorithms, and validate its performance using the Malaga instance as the test instance: the automatic migration tuning technique. This technique, developed based on a theoretical study on the effect of migration on the convergence in a distributed algorithm, consists in determining at which moments (i.e., after which iterations) the migration process should take place. We propose a configuration for the technique and its embedding into a distributed GA. The experimental results show that the proposed technique obtains results of quality comparable to the those of the best found fixed migration schedules, requiring a similar computation time, and achieving only slightly lower quality than the best found equivalent sequential execution. Additionally, our proposed technique can alleviate the burden of parameter tuning for the migration process configuration, and effectively reduces the whole optimization process time by at least 75%.

Part III

WIRELESS SENSOR NETWORK DESIGN

Chapter 7

Wireless Sensor Networks Layout Optimization

We describe in this chapter the second fundamental problem found in Wireless Sensor Networks that is tackled in this thesis, namely the Wireless Sensor Network Layout optimization problem (WSNL). This problem has a similar starting point than the RND problem, as the purpose is to obtain a network that produces a high coverage of a terrain (this can either be an optimization objective, or a constraint); however, unlike the previous one, it is not a combinatorial problem where locations are selected from a pool, but a continuous optimization problem where locations are freely chosen (although generally discretized). Furthermore, WSNL takes into account the ad-hoc communication network of the WSN, which was not considered in RND, and finds a layout that not only will produce a good coverage, but whose respective communication network has certain desirable properties.

Therefore, the philosophy behind WSNL is not the same that was behind RND/scheduling. This problem is much more scenario dependent than RND. In WSNL, the designer assumes it has absolute control over the nodes positions in the field. This assumption does not exist in the RND/scheduling problem, where nodes are already deployed prior to the schedule design, and there is no control over the deployment process. Thus, the two problems can be regarded as complementary on the network deployment conception: when one can decide single node's locations, the WSNL is defined (and resolved) to decide the best possible locations with regard to the desired network properties, when one cannot decide single node locations, a schedule problem is defined (and resolved) to select the best fit set of working nodes, again with regard to the desired network properties.

Another difference between WSNL and scheduling concerns the economy in terms of the number of nodes. One of the principles behind the scheduling is the assumption that the network contains an excess number of nodes; hence, only a subset of the nodes need to be active at a time to provide the desired levels of coverage (or quality of service, in a broader conception of the network service) and a connectivity structure (when connectivity is an issue). This assumption no longer holds for WSNL. In fact, one of the optimization objectives in WSNL (though admittedly not the one with the highest priority) is to minimize the economic cost of the network by minimizing the number of nodes (which cost money). Thus the number of nodes is not *in excess*, but *tailored* to fit the network requirements.

In this chapter we will first provide a general description of the WSNL problem, and the models used in the existing literature for the coverage and the communications of the sensor node, and the sensor network, as they both affect the operating features of the network. We will also discuss the lifetime computation, since it constitutes another of the objectives. Finally, we provide a review of the literature of the field.

7.1 Problem description

The Wireless Sensor Network Layout optimization problem (WSNL) is widely considered one of the fundamental problems in WSN ([166]). In its most basic form, WSNL amounts to selecting the geographic locations for the deployment of each single node of the network.

There are two main concerns in the WSNL problem: coverage and connectivity. The coverage amounts to the basic quality of service offered by the sensor network, and it has to be maximized. The connectivity makes reference to the communication topology resulting from the node positioning. The main goal sought in the topology, besides the hard constraint of the network being fully connected with the HECN, is that the communication structure is such that the energy consumed for communications is minimized, hence maximizing the lifetime of the WSN. Additionally, the economic cost of the network (generally, the number of sensor nodes employed) is set as a third objective in WSNL. The number of nodes and connectivity can in principle be considered as independent objectives; however, we will see that they are in fact opposing objectives.

Some forms of the WSNL problem have been demonstrated to be NP-complete ([218]). Additionally, the WSNL can be reduced to the set covering problem, by restricting the available positions of the sensor nodes to a set of discrete locations (for instance a regular point grid); and the set covering problem is well known to be a NP-complete problem ([42]). Therefore, we state that the WSNL is NP-complete as well. For this reason, metaheuristics seem an adequate tool to tackle instances of this problem of large size.

7.2 Models employed for the coverage

One of the fundamental elements involved in the WSNL problem is the model employed for the coverage, since coverage is the main optimization objective (or constraint). We can establish several classifications of the different coverage models that have been used in this domain.

7.2.1 Node coverage models

The first classification is made according to the individual node sensing model:

- Binary coverage ([20, 106, 151, 190]): the node fully covers a disk of radius R_{SENS} centered at the node location. When the distances are normalized so that $R_{SENS} = 1$, the model is also referred to as Unit Disk Coverage model (UDC).
- Probabilistic coverage ([213]): the node covers a disks of radius R_{SENS} centered at the node location, but points inside the disk are only covered with probability $k < 1$. This value is also referred to as *detection probability*.
- Quasi Unit Disk (QUD) ([61, 218]): a distance dependent combination of the previous two models. The node gives full coverage to a small disk of radius $\alpha \cdot R_{SENS}$ where $\alpha < 1$, and probabilistic coverage to the crown defined by $\alpha \cdot R_{COMM} < r < R_{COMM}$, with probability $P = f(r - \alpha \cdot R_{COMM})$, where the function f must meet the following description:

$$f : \Re^{+} \rightarrow [0;1], \left\{ \begin{array}{rcl} f(0) & = & 1 \\ f(1-\alpha) & \geq & 0 \\ f'(x) & \leq & 0 \text{ for } x \in [0; 1-\alpha] \end{array} \right. .$$

  A typical function used for the QUD model is the decreasing exponential function:

$$f(d) = \exp(-A \cdot d),$$

  where A is a constant value used to tune the decrease rate of the detection probability.

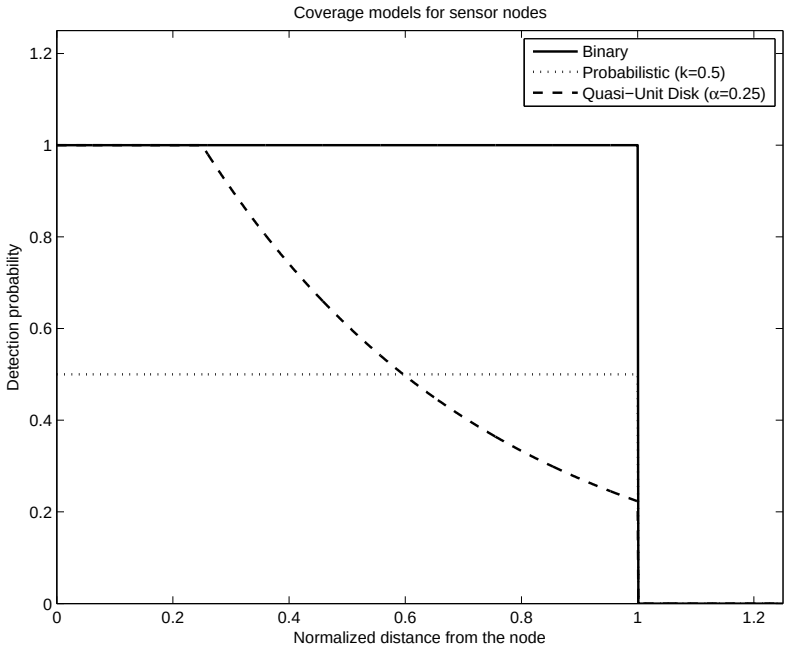


Figure 7.1: Coverage models of a sensor node: (a) binary, (b) probabilistic, and (c) quasi-unit disk

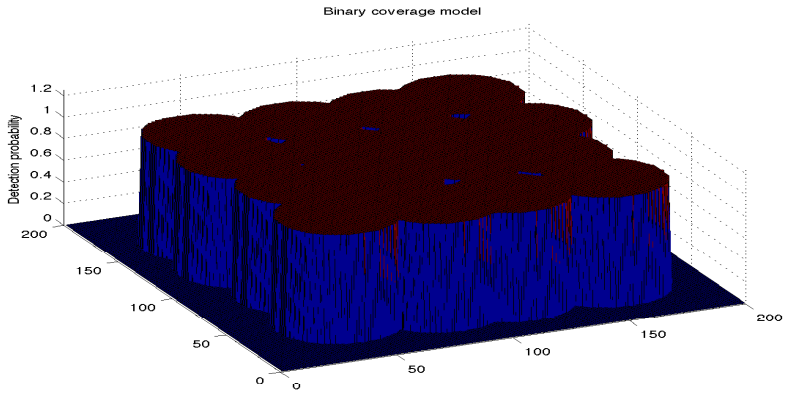
- Other coverage models: sometimes the coverage model is specific of the application at hand and does not fit into a general category. Some examples are directional coverage (the sensors point at a given direction and can only sense events inside a given angle and distance bounded region, [2]), or boundary sensors ([231]), that are sensors that define a "barrier" and detect when an object traverses the barrier (much like the laser traps in spy movies).

The three first node sensing models are illustrated in Figure 7.1, by the detection probability as a function of the normalized distance of the event (or object) to the node. The normalization is made to R_{SENS}. The effect of these different node models when applied to a network coverage are shown in Figure 7.2. A WSN is deployed as a perturbed regular grid (i.e., the nodes are placed forming a regular $N \times N$ square grid, then each node has its position slightly displaced from the grid point), then the detection probability for every point in the sensor field is calculated and plotted using the binary model (Figure 7.2a), the probabilistic coverage model (Figure 7.2b), and the quasi-unit disk model (Figure 7.2c). In the models plotted in the figures, the probabilistic coverage is set to $k = 0.5$ and the quasi-unit disk model is defined by a decreasing exponential with $A = 2$ and $\alpha = 0.25$. As it can be seen, when the binary model is used, the resulting network coverage is binary as well (Figure 7.2a); when the probabilistic model is used a set of "plateaus" appears (there are four different detection levels in Figure 7.2b); finally, when the quasi-unit disk model is used, the resulting network detection probability is a continuous value (Figure 7.2c).

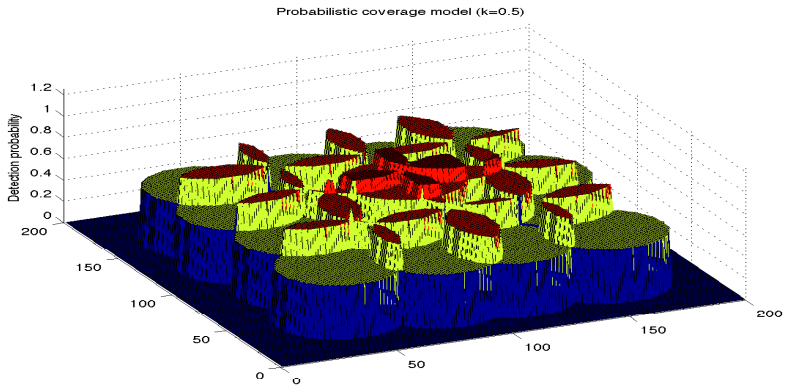
7.2.2 Network coverage models

A different classification may be made according to the field concept, that is, the method by which the coverage of the *network* is evaluated, as opposed to the coverage offered by a single node. This depends mainly on the purpose of the WSN, that is, what and how is the network supposed to be monitoring. This classification is partially related to the one described above. The main cases one can find are the following:

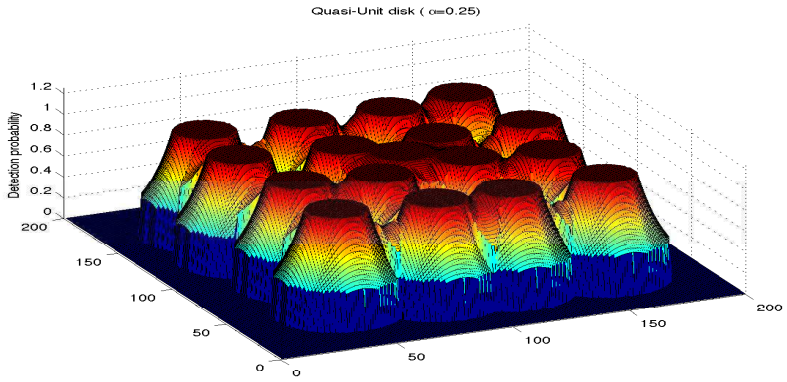
- Point coverage model ([42, 106]): the network administrator is only interested in monitoring a dis-



(a) binary coverage



(b) probabilistic coverage



(c) quasi-unit disk coverage

Figure 7.2: Network coverage for different sensor node coverage models on a ground 2D terrain: (a) binary, (b) probabilistic, and (c) quasi-unit disk.

crete set of points (this recalls the coverage definition for the AFP problem in Section 5.2.1). This model can be combined with any node coverage model if the points are physical points in the terrain and their distances to the nodes can be calculated.

- Area coverage ([20, 105, 106]): the network administrator is not interested in particular points, but the whole area of the sensor field. This is, by far, the most common assumption, not only in WSNL but in many other problems defined in WSN (scheduling, etc.). There are still several combinations that can be used within the general assumption of area coverage. When area coverage is used with binary node coverage, the evaluation of the coverage is the total area that is at least covered by one node. When probabilistic or quasi-unit disk node coverage are used with area coverage, either the average detection probability (integrated over the whole area of the sensor field) or the total area where the detection probability surpasses a given detection threshold may be used, depending on the formulation and requirements of the problem at hand.

- k-coverage ([35, 96, 133]): this one is a simple extension of the area coverage, but where k sensor nodes are required to cover any location point (this redundancy can be used for enhanced robustness versus individual node failure, for additional information, or to improve the false alarm or missed detection rates); a point that is covered by k different sensor nodes is said to be k-covered. The k-coverage assumption is only used with binary coverage models (not probabilistic), and the number of nodes that can cover a given terrain point is known as *coverage degree* of that point. A particularly interesting instance is the 3-coverage, since 3 is the threshold for localization by trilateration, therefore solving the WSNL using the 3-coverage assumption produces a WSN that can perform trilateration and return the location of the sensed events. The evaluation of the network coverage can be based on the calculated average coverage degree over the complete sensor field (to be compared to k), or on the total area where the coverage degree is at least k.

- Differentiated coverage ([61, 105, 224]): this is the generalization of the area coverage assumption. In differentiated coverage, the same coverage requirements do not hold for the whole field. Instead, there are some parts in the field that are considered critical (of higher importance), and as a result require a high coverage, while some other parts are less important and require lower coverage. The specification of coverage degree requirements can be very simple (with as few as two different levels), or arbitrarily complex. Differentiated coverage can be used in combination with either binary node coverage (then the coverage degree is used as the varying coverage requirement), or probabilistic coverage (then the detection probability is used as the varying coverage requirement). The evaluation of the coverage corresponds to the total area where the requirements are met, which can be further weighted in such a way that area with high requirements receive a large weight and area with low requirement receive a low weight (the coverage requirement can be used as the weight factor).

- Perimeter coverage ([25, 102]): in this scenario the network administrator is no longer interested in covering a full terrain, but just the boundary of that terrain. This assumption is used in WSN designed for intrusion detection. This model can be used in combination with any node coverage models. The evaluation of the coverage may correspond to the percentage of the perimeter zone that is covered by at least one sensor, the percentage of the perimeter that is k-covered, or the percentage of the perimeter where the detection probability surpasses a given threshold value.

- Path coverage ([25, 134, 151, 190, 206]): in this case the node administrator wants to detect moving targets inside the sensor field. Therefore, the coverage of the network is not evaluated over terrain points, but over paths, that is, lines inside the sensor field. Paths can enter and leave the field, or can be originated and terminated in points within the sensor field; generally, a minimum path distance is assumed (to make the problem tractable). Path coverage is used in combination with probabilistic or quasi-unit disk models, and the probability of detecting a given path is calculated by integrating the detection probability over the path, assuming a given target speed and a time dependence of the

detection capability of a node. The evaluation of the coverage corresponds to the path detection probability, assuming some nature and probabilistic distribution of the paths.

- Hybrid or multi-nature coverage: this assumption is used when there are several physical magnitudes that are sensed, and several kinds of sensor accordingly. This can be seen as the generalized problem for an arbitrary number of sensed data, with different requirements. An example of a system that adheres to this definition is an environmental WSN used in the forests of California; each sensor node is equipped with light, temperature and humidity sensors. Each type of sensor has its own coverage model, and for each kind of measured parameter different coverage requirements may be defined.

In our definition or the WSNL problem (Chapter 8), we use **binary coverage** for the sensor node and **area coverage** for the sensor network.

7.2.3 Computation of an area coverage

Several methods have been proposed for the estimation of the coverage of a given terrain area by a set of nodes ([61]). In this section we offer a short review of the most frequently found methods in the literature. These methods are:

- Use of superimposed regular point grid.
- Mathematical analysis.
- Use of Voronoi diagrams.
- Check the intersections among sensing disk boundaries.

The simplest is the definition of a superimposed regular point grid, which can be viewed as a generalization of the point coverage assumption, where point coverage meets area coverage ([2, 3, 61, 105, 217, 218, 224]). Each point in the grid has an associated terrain area around itself, the coverage is estimated for each point in the grid and all the corresponding area is considered to have that coverage value. The grid can be used to compute either coverage degree or the detection probability. The grid is an approximate estimation of the real coverage, but the approximation can be made as accurate as desired by increasing the number of grid points (hence reducing the area per point), at the cost of more computational effort (the computational effort of this method is $O(n)$ with respect to the number of grid points, or $O(n^2)$ with respect to the accuracy). Figure 7.3a shows an example grid computation of the coverage; every node has to compute which of the grid intersection points fall within its sensing disk.

A mathematical analysis of the coverage can also be used, however, the complexity of it becomes unmanageable as the number of nodes increase and thus this kind of method is generally employed only to analyze regular node deployments ([20, 70, 106]).

The use of the Voronoi diagrams, and their counterparts the Delaunay triangulations, is very popular in WSNs for coverage and connectivity purposes ([17, 52, 70, 80, 136, 140, 219, 234]). Voronoi diagram is a special kind of decomposition of a metric space determined by distances to a specified discrete set of objects in the space, e.g., by a discrete set of points (also called a Voronoi tessellation, a Voronoi decomposition, or a Dirichlet tessellation, it is named after Georgy Voronoi). In the simplest case, which is the one that applies to WSNs, we are given a set of points S in the plane, which are the Voronoi sites. Each site s has a Voronoi cell $V(s)$, consisting of all points closer to s than to any other site. The segments of the Voronoi diagram are all the points in the plane that are equidistant to the two nearest sites. The Voronoi nodes are the points equidistant to three (or more) sites. For a WSN in which the Voronoi diagram has been defined, a sufficient condition for coverage is that every node entirely covers its Voronoi cell. Figure 7.3b shows an example computation of the coverage with Voronoi diagram; every node has to check whether its sensing

disk covers its Voronoi cell entirely, which can be reduced to check whether all the nodes of its Voronoi cell fall within its sensing disk. Therefore the complexity of the method is $O(n)$ where n is the number of neighbors of a node[1]. This technique can be generalized for k-coverage.

Another method for estimating global k-coverage of a terrain was proposed by Wang *et al.* ([209, 219]). The method assumes for simplicity that nodes cover all the area in their sensing disks except the boundary. Then, the intersections between the boundaries of the sensing disks are all checked in terms of coverage (note that none of the nodes that produce the intersecting sensing disks is taken into account for the coverage computation), as well as all intersections between the boundary of a coverage disk and the boundary of the terrain field. If there are such intersections, and all of them are k-covered, then the whole sensor field is k-covered as well. Figure 7.3c shows an example application of this technique; the intersection points are marked with small empty disks.

The two latter techniques are computationally more efficient than the use of the grid, but present a big drawback as well: they produce only a binary output, that is, either *the terrain is fully (k-)covered*, or *the terrain is not fully (k-)covered*; in the second case it does not provide any additional information (like, for example, the percentage of the terrain which is (k-)covered or the average coverage degree of the sensor field). One could always use the ratio of intersection points or Voronoi nodes (depending on the case) that receive coverage as a reference, but there is no guarantee that these points are evenly distributed throughout the sensor field and that henceforth this ratio is a significant value in the sense that it provides a trustful indicator of the area coverage degree. Therefore, since metaheuristic optimization techniques require evaluation functions that act as guiding functions in the suboptimal regions o space, the latter coverage estimation methods are not suited for our purposes. Thus, the **grid evaluation method** is the one selected to be used in our calculations.

7.3 Models employed for the communications

Connectivity is, besides coverage, the other big issue in the WSNL problem. From the description of the operation of a WSN (Chapter 2), a node that does not have a communication path with the HECN is considered disconnected, and thus its coverage is not taken into account for the computation of the WSN coverage. Therefore, it is important that the WSN produced in WSNL is a connected network, and that it is connected to the HECN as well.

7.3.1 Link level

The general model for connectivity is most similar to the coverage model, and defines connectivity based on the range parameter, also known as the communication radius R_{COMM}. Depending on the distance between two nodes and the value of R_{COMM}, these nodes will have a direct communication link or not. However, there are many variations of the model:

- Unit disk model ([98, 106, 140]): Similar to the unit disk model for coverage. Two nodes separated by $d \leq R_{COMM}$ have a communication link, two nodes separated by $d > R_{COMM}$ are out of communication range of one another and thus do not share a direct link. The resulting topology is often called a *unit disk graph* (UDG).

- Probabilistic link ([151]): Expands the unit disk model by adding a probability of error E, such that two nodes separated by $d \leq R_{COMM}$ have a link with probability $P = 1 - E$. The link probability is evaluated once per link: either the link exists or does not exist, for the whole WSN lifetime.

- Quasi-unit disk ([26, 233]): Similar to the coverage equivalent. Two nodes separated by distance $d \leq \alpha \cdot R_{COMM}$ with $\alpha < 1$ have a link, two nodes separated by $d > R_{COMM}$ do not have a link. If

[1] This complexity is *per node*, and without taking into account the complexity of determining the Voronoi cell of a node.

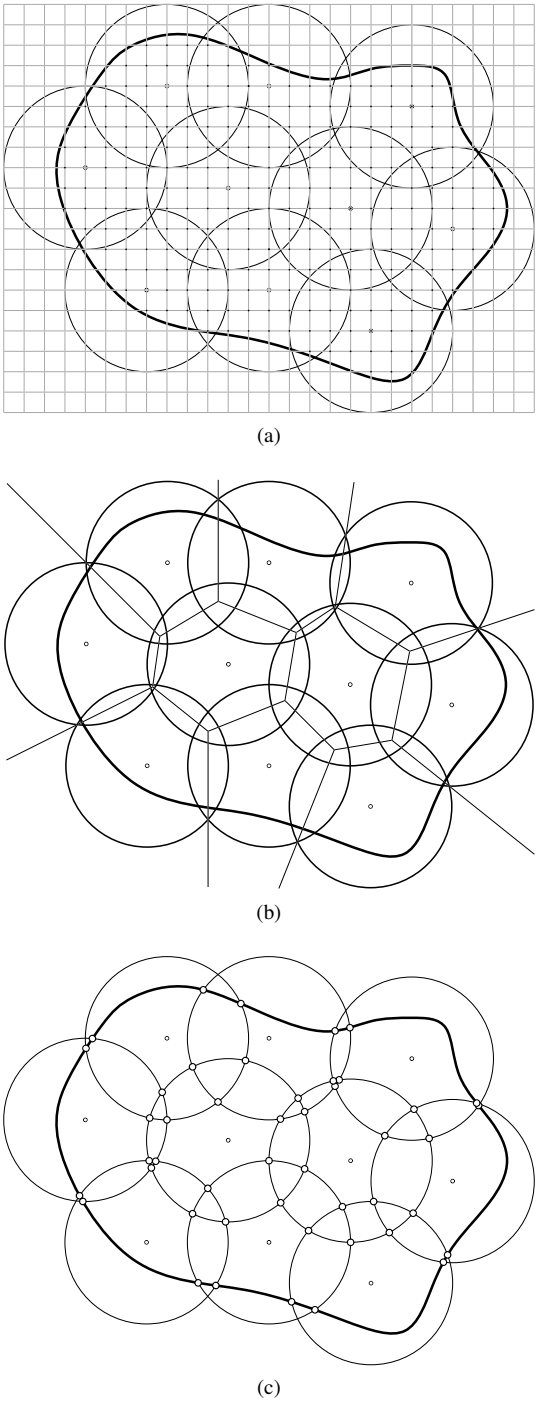


Figure 7.3: Methods for area coverage computation: (a) superimposed grid, (b) Voronoi diagram, (c) sensing disks intersections.

the nodes are separated by $\alpha \cdot R_{COMM} < d < R_{COMM}$ they may or may not have a communication link (possibly with some probability).

There is an alternative model, in which the link distance is unbounded (or equivalently, $R_{COMM} = \infty$, [42, 188]). In this model, all communication links exist and may be used by the nodes; either the network degenerates into a 1-hop network (since there is no link distance limit, all nodes are within communication range of the HECN directly) –and the bandwidth problem arises since all nodes cannot be transmitting at the same time, or else they would interfere with each other ([188])–, or it remains multi-hop. The main reason why such a network would not degenerate into a single hop network is because of energy efficiency. In this model (as well as in many models where the link distance is bounded by R_{COMM}), communication links have an associated communication power of the form $P = P_0 + K \cdot d^{\alpha}$, where $2 \leq \alpha \leq 5$ is the path loss exponent ([40, 188, 199]). This power figure makes short communication links more energy-efficient than long links, so they are generally preferred.

7.3.2 Network level

Very similarly to what happened in the case of coverage, the WSN may require single connectivity (that is, there exists at least one path between any node and the HECN), or k-connectivity (there are k disjoint paths between any node and the HECN, [134]). The latter is used to enhance network robustness: the network does not become disconnected even if *any* $(k-1)$ nodes fail. Then, several classifications can be made about the network communications: according to communication model, according to the establishment of the topology, according to the routing algorithms used, etc. We will briefly review these classifications below.

The communication model states where the communications originate, and where they end. In other words, the communication model defines which nodes will communicate with which other nodes. Basically, there are two communication models employed in WSNs:

- N-to-1: This is the simplest and most widely used model in WSNs. In it, every node can only communicate with the HECN, and all other nodes can only be used as relays for the multi-hop communication.
- N-to-N ([144]): This model corresponds to a fully ad-hoc network. In it, all nodes are in principle susceptible of sending information to any other node. This model (or a subset of it) is used when nodes can autonomously perform some operations, or perform local data fusion & processing: these operations do not require them to necessarily contact the HECN, but instead communicate with geographically close neighbors.

The topology of the network states mainly which links exist and which links do not. All links selected by the topology are valid links according to the sensor node link model (Section 7.3.1), that is, only links that exist from the link model's viewpoint can be selected for the topology. Nevertheless, there are links that could exist from the link model's viewpoint, but are not selected by the topology; the WSN operates as though these nodes do not exist. We can find the following models for the network topology (illustrated in Figure 2.3):

- **Plain network** or flat network, also known as ad-hoc network. All nodes are considered equal, therefore all possible links (depending on the link level conditions) are set and may be used.
- **Hierarchical** (clustered, [92]). The nodes are organized as a two-tiered (or k-tiered) network. At the first tier are "regular" nodes which communicate only with a single node of the higher tier. Nodes of the second tier (and above if more tiers are defined) are *cluster heads*, they are connected to all the regular nodes inside their cluster, to other cluster heads, and possibly to the HECN. Therefore, only the feasible links that adjust to the network topology may be used.

Last comes the routing within the network; the routing corresponds to the selection of links from the topology in order to relay some data from a source node to its destination (HECN or some node). Routing constitutes one of the most important issues in WSNs and receives a large deal of attention in research (see Chapter 2), however it is out of the scope of this thesis, hence we will only give a brief review of it. Regarding the routing models used in WSN, we can find the following:

- Shortest path ([66]). This is the basic routing algorithm, where each data packet simply traverses the network following the shortest path from its originating node to the HECN. The concept of shortest relies on the minimal hop distance, and generally requires a previous gradient generation by a special broadcast from the HECN, and any other node that may be the destination recipient to some information. Obtaining the global shortest path for every node requires the use of routing tables at the different nodes, which can be too high a cost to be affordable, especially for large WSNs.

- Geographic Forwarding (GF, [228]). Similar to the previous, tries to look for the shortest path. However, instead of the global hop-count to the HECN, GF relies solely on the local geographic locations of the nodes (and the HECN). The chosen approximation used is a local greedy rule, where at each step the relay node chooses among its neighbors the one that is closest (geographically) to the HECN.

- Face Routing (FR), or Greedy Perimeter Stateless Routing (GPSR, [26]). In this routing, the message is forwarded in such a way that it traverses the set of polygons defined by the WSN topology clockwise or counterclockwise (the *faces* of the polygons, hence the name). This method avoids getting stuck in local optima (that is, nodes that have no neighbor geographically closer than themselves to the HECN) as the greedy algorithm does, hence a combination of both has been proposed to escape the impasse reached by the greedy.

- Energy-aware routing systems ([101]). These routing models use energy-efficient paths with the global aim of maximizing the network lifetime. There are two approaches for this: choosing the path with lowest energy expense, or choosing the path with minimal rate of required energy to remaining energy. A global algorithm to choose such a path is, for example, Dijkstra's algorithm. Again, finding the global optimum is generally unfeasible, hence local greedy rules are often employed: choosing the neighbor node (closes to the HECN) that requires minimum link energy, or choosing the neighbor with minimal rate of link energy to remaining energy.

- Local energy-balancing heuristic (see Section 8.1). We propose this local routing algorithm as a lightweight specific energy-aware routing algorithm. In it, every node detects all the neighbors that are closer (in hop count) to the HECN. Then, all the traffic relayed through it is distributed among all those neighbors; the distribution may be even (all neighbors receive the same data amount), or energy-dependent (each neighbor receives an amount of data inversely proportional to its link energy). This algorithm is aimed at balancing the energy expense among nodes, in order to reduce the bottlenecking in the WSN.

- Specific routing algorithms. There are many routing algorithms specifically proposed for WSN, but this subject is out of the scope of our work. A short review can be found in Section 2.5.

7.3.3 Additional considerations

Connectivity is not always an issue in the WSNL problem, it is sometimes overlooked by assuming that $R_{COMM} > 2 \cdot R_{SENS}$. It was proved in [209] that, when that inequality holds, coverage implies connectivity. Therefore, by assuming this ratio between the radii holds, the WSNL is reduced to ensuring the coverage.

Nevertheless, still communications remain one of the most important issues in WSNs, for they are widely assumed to determine the dominant part of the energy consumption budget during the operation time of the WSN (in contrast, the energy spent in computation is often considered to be negligible). Therefore, in order to maximize the lifetime of the network (Section 7.4), one must optimize its communication structure.

In our problem definition, communications are definitely an issue. We employ **binary coverage** at node level, assume a **N-to-1** communications model over a **flat** network structure; the routing mechanism is the **local energy-balancing** heuristic routing.

7.4 Lifetime in WSNs

The lifetime is defined as *the duration of a thing's existence or usefulness*. In WSNL, as in most WSN design problems, lifetime corresponds to the operation time of the system, and is one of the optimization objectives.

The calculation of the lifetime of a WSN is not a simple issue. Unless the operational requirements of a WSN are clearly stated, in which case the first moment they are not met is considered the end of the network lifetime, the exact moment where the WSN is considered to stop operation is fuzzy. Generally, the nodes in a WSN will gradually stop functioning due to energy depletion. There are many definitions of lifetime, but most of them are specific to a given WSN application or problem. The most widely used definitions for general-purpose WSNs are:

- **Time To First Failure** (TTFF, [35, 101, 192]). This is the simplest criterion for the calculation of the lifetime: the moment the first node runs out of energy is considered the end of the lifetime of the network. Under this assumption, it suffices to identify the network energy bottleneck, the node that spends the highest energy per time unit, to calculate the network lifetime (for heterogeneous networks, it is the node that spends the highest energy with respect to its total energy per time unit instead).

- **α-lifetime** ([229]). This definition is less restrictive than the previous one; in this case, the lifetime is the moment the network offered coverage falls below an α ratio of its initial value due to nodes running out of energy.

- **Connected network** ([35]). In this definition, the lifetime lasts until the WSN becomes disconnected, that is, there are nodes that become separated from the HECN while they still have remaining energy.

For our problem definition, we will use the **TTFF** criterion.

7.5 Literature review

There are many works in the literature that tackle the Wireless Sensor Network Layout problem. Interesting surveys on coverage problems defined for WSNs, that are mostly related to our defined WSNL, and previously presented scheduling problems, can be found in [34, 226, 202]. Specifically, in [226], the authors classify node placement problems into two categories: static and dynamic. Our work belongs to the first category. Different works use different approaches to the problem, make different assumptions, set different optimization objectives, and use different models for the problem, the network, and the sensor nodes. Among the most popular approaches, we can find:

- Tackle it as a combinatorial optimization problem: select which nodes have to be activated from a set of deployed nodes. Put in other words, define it as a scheduling problem.

- Assume that nodes follow a random deployment.

- Define the problem as a continuous optimization problem. Select the location of the nodes to be deployed.

- Use a regular geometric deployment.

- Rely on geometry-based computation: Voronoi diagrams and Delaunay triangulation.

Most of the early work on node deployment assume that nodes cannot be placed deterministically, but occupy random positions instead. This line of work usually follows one of two leads: in the first, the authors assume a given distribution function and get the resulting performance statistics from the network (usually, expected values, and upper/lower bounds); in the second, the distribution function can be optimized (for instance, a parametric function may be defined) so that the resulting network has the best possible performance statistics. The expected coverage achieved by random node deployments with homogeneous distribution is studied in [133]. The authors are interested by three types of coverage: binary area coverage, k-coverage, and k-*least*-coverage (meaning that k or more nodes are covering the area). The study is based on the problem's similarity with the set intersection problem, and the probabilities that a single point is single, k- or k-*least*-covered are obtained. Finally, the authors indicate the procedure that should be taken to study the same problem in the case of a non-homogeneous distribution of the nodes. In [151], the authors consider the coverage of a straight path by randomly deployed nodes. The nodes follow a Laplacian distribution, but a non-homogeneous one (with $\lambda = f(x, y)$); additionally, the nodes have random values in $[0; 1]$ for R_{SENS}. The probability density of k-coverage for a path is obtained as a function of its length. Since no closed forms are obtained, lower and higher bounds are derived. In [190], target tracking with randomly deployed binary sensors is studied. The accuracy of the tracking depends on the partition made of the space by the sensing disks, the largest the diameter of these parts, the lowest the tracking accuracy. The upper bound on spatial resolution is derived, and found to be in the order of $1/(\rho R_{SENS})$, where ρ is the node density by area unit. By detecting at which moments the target crosses from one patch to another, the target's speed may be estimated with some accuracy as well. For this, the authors propose OccanTrack, an algorithm that searches for the path with the minimum number of straight segments that cross the boundaries between regions in an orderly fashion, with a constructive greedy process; assuming that trajectories consist in fact of straight segments, this algorithm is proved to produce the least square error. Finally, a particle filtering algorithm with geometric postprocessing is proposed to handle the case with quasi-unit disk coverage from the nodes. In [35], the utility, coverage, and lifetime of a randomly deployed WSN of acoustic nodes are optimized. The WSN has to be designed fulfilling a budget constraint, by selecting the sensor nodes that will constitute it. Four approaches are considered to solve the problem: a full multi-objective conception, the optimization of a single objective, optimize a scalar fitness function that combines the different objectives (mono-objectivization of the problem), and optimize a single objective while imposing constraints on the rest. An incremental algorithm that seeks Pareto-optimal solutions, as well as a continuous relaxation of the formulation are proposed to solve the problem. Node deployment strategies for object detection in the 2D plane are proposed in [25], considering that both objects and nodes may be static or mobile; when nodes are mobile, the movement coordination among nodes is also analyzed. The node densities for obtaining given detection probabilities are stated. Detection probabilities for random node deployments (following a Poisson distribution) as well as random movements are also studied. The same study is then made for finite sensor fields, using similar deployment strategies except for the mobile objects scenario, in which nodes are deployed to obtain perimeter coverage. The position of the nodes of a WSN is designed to maximize the detection probability of moving objects in [213]; as a novelty, the authors optimize the *node density* function (built as the sum of weighted Gaussian functions centered at the points of a regular grid), arguing that the obtained solution is therefore independent of the size of the WSN and thus scalable. To avoid false alarms, k-coverage is required; thus k nodes have to be within a strip of half-width R_{SENS} drawn around the axis made of the object path. They propose a two-phase resolution process, with an initial GA that produces a rough guess, followed by a refining phase with a Sequential Quadratic Programming (SQP), to

solve four different scenarios (with different configurations of the object's movement). Finally, the authors propose a sampling method to obtain the locations of the nodes of a WSN from a node density function.

Regular or systematic node deployment strategies have also been researched, as they present the advantage of simplicity and scalability. In [20], the authors study different regular deployments to guarantee full coverage to the sensor field, using binary coverage model. They compare square, triangular and hexagonal lattices and obtain the required node densities for each case depending on the values of R_{SENS} and R_{COMM}. Fault tolerance is also checked, considering the minimum number of node failures that disconnects the network; two systems to increase fault tolerance are proposed. The hexagonal regular lattice is also studied in [70], where the authors propose a superposition of two lattices to form a robust network that offers 2-coverage, and indicate the relative positions of nodes of one lattice with respect to the other. Connected coverage of the 2D plane, of a planar concave sensor field, and of a set of points, by *systematic node deployments* are studied in [106]. The authors assume that $R_{SENS} = R_{COMM}$, and propose two methods. The first covers the 2D plane and the planar concave sensor field: they cover the region with connected rows of nodes separated by the maximum distance possible while keeping the coverage tight (there are no uncovered holes); then they add an extra column of nodes that connects every pair of consecutive rows (that were previously disconnected). For the point coverage, they generate a set of points $\mathcal{C}$ that contains said points; at each iteration, they place a node in a point in $\mathcal{C}$, remove all points covered by the placed node from $\mathcal{C}$, and add all points corresponding to the intersection points between the coverage perimeter of the node and the coverage perimeters drawn around the points remaining in $\mathcal{C}$. The process is repeated until no points remain in $\mathcal{C}$. Regular node deployments are studied for different ratios of R_{COMM} and R_{SENS} in [17]. The authors prove that square, triangular, hexagonal and rhomboidal regular deployments are all optimal for different ratios of the sensing and communication radii (full connected coverage is always maintained). However, the most efficient technique (its asymptotic optimality is demonstrated) is the strip-base regular deployment, where connected node strips are piled up to obtain 100% coverage, then an extra strip is placed orthogonally to provide connectivity (if necessary), or two extra strips for 2-connectivity. In [231], node placement and detection probability using boundary nodes (nodes that define a $n-1$ "barrier" in a n-dimensional space and detect whenever an object goes through the barrier) are studied. For 1-dimensional WSNs the uniqueness of node detection sequence with respect to the path of the object is studied. For 2-dimensional WSNs, a *regular triangular barrier lattice* is proposed that detects the object's location after it traverses two frontiers. A conception of the WSNL problem that is halfway between the use of a regular structure and the scheduling problem is considered in [217]. Two regular geometric lattices are proposed for sensor nodes with adjustable R_{SENS}. In the first one a coarse hexagonal lattice with no overlap among sensing disks is formed with nodes using large R_{SENS}, then the resulting coverage holes are filled with nodes using small R_{SENS}; in the second, the same hexagonal lattice is used for nodes with large R_{SENS}, but the holes are covered with two kinds of nodes, with medium and small R_{SENS}. The energy consumption is proportional to R_{SENS}^k, with $k = 2$ or $k = 4$. The analogy to scheduling is made by assuming that the deployed WSN is very dense, and that nodes can be selected close enough to the locations of the proposed lattices. Lifetime is the main concern in [98]. The authors first assume that nodes follow a given schedule (modeled by a stochastic process that assigns working and sleeping probabilities to the nodes independently), and the individual node lifetime probability density function is Gaussian. Then they derive the lifetime probability density functions (PDFs) for networks employing *square and hexagonal lattices* for node deployment. The authors assume that $R_{COMM} = R_{SENS}$.

However, our work is focused on non-systematic deterministic node placement. In this field we can find a very large body of research knowledge, which can be mainly classified into two types, regarding the resolution methods employed. The first group includes works that use specific methods, often referred to as *ad-hoc heuristic* methods, tailored after the specifics of the problem instances at hand. A recurrent case is the use of greedy methods. In [61], a regular grid is used to compute the detection probability of a WSN and to place the nodes in order to obtain differentiated coverage. The authors propose two *greedy strategies* for the node deployment: the first one places a node at each step in the position that maximally reduces the accumulated probability of non-detection, the second one places a node at each step in the

position with minimal detection probability. By adding a negative bias that depends on the distance d between consecutive points in the grid to the computed detection probabilities, the authors correct the error introduced by the grid model approximation of the terrain. Connectivity is not considered in this work as an issue. Zhang and Wicker ([230]) study the positioning of sensors in a terrain from the point of view of data transmission. They divide the terrain into cells, then analyze how N sensors should be distributed among the cells, in a way that avoids network bottlenecks and data loss. An *ad-hoc heuristic* algorithm is proposed for node distribution. In [80], the deployment of the nodes to reduce the distortion and the energy consumption (due to transmissions) is studied. Two codification systems for the data, joint-entropy and Slepian-Wolf, are considered. The distortion is considered to be relative to the maximum distance between any sensor and its farthest sensed point, according to a Voronoi partition of the sensor field. The problem is solved for one-dimensional WSNs, and an *ad-hoc heuristic* solution based on concentric circles is proposed for two-dimensional networks. A sensor placement for perimeter coverage is presented in [102], with the purpose of detecting a moving agent. The field is assumed convex, and the moving agent has to be detected as it enters or leaves the field. Given the assumptions and supposing that nodes may only be placed in the perimeter, a node's position can be uniquely identified by its angular value θ with respect to a central reference inside the field. The Position Error Bound (PEB) as a function of the angle is obtained, and a *greedy* method that iteratively selects the angle whose value minimizes the PEB, by performing a complex coordinate transform, is proposed. The study is then generalized to include weighted nodes and multiple moving agents. An estimation of the detection of moving targets by a WSN is given in [134], along with a *node deployment strategy*. Based on the analogy with the line set intersection problem, the detection probability is obtained for a single node, and it is found to depend only on the perimeter of its coverage. Detection probabilities for WSN with high number of sensors are difficult to compute, hence lower and higher bounds are proposed. The proposed deployment strategy, DATE, seeks to maximize the internode distance so as to minimize the overlap between coverage cells; it achieves so by solving the circle packing problem. The connected version, CDATE, deploys nodes iteratively in descending R_{COMM} order, ensuring that starting from the k^{th} node, all nodes are k-connected. A set of BSs for node location purposes has to be selected from a pool of deployed nodes in [174]. This problem is halfway between WSNL, scheduling, and Location Discovery (LD). The basic idea is to divide the network in as many regions as possible, where for every region pair there is one BS that can discriminate with low error probability using the received signal from the new node. For this, the Generalised Likelihood Ratio Test (GLRT) is used in combination with a family of PDFs to increase the robustness. When a new node appears, an iterative pairwise comparison between regions (in ordered fashion) can state in which region that node is. The more regions one can define, the smaller each region is, hence the smaller the uncertainty in the node's assigned location. Lifetime is also the main concern in [40], but instead of raw lifetime, they study the lifetime per node, that is, the ratio between the network's lifetime and the number of nodes in the network. They restrict the study to one dimensional WSNs where the HECN is located at the top left, and the transmissions are multihop, with every node communicating strictly with its immediate neighbors. The energy consumption takes into account the sensing energy, the message reception energy, the transmission constant and distance dependent energy (which depends on the distance d in the form of d^λ, $2 \leq \lambda \leq 4$). The authors propose a *greedy algorithm* for node placement along the WSN axis, and from it derive the optimal number of nodes and their positions.

The second group includes the works that use general-purpose flexible optimization methods, namely metaheuristics. This body of research contains a high number of publications, among which we select the ones that tackle problems resembling our WSNL problem. Jourdan and de Weck solved an instance of WSNL using a multi-objective genetic algorithm in [101]. In their formulation a fixed number of ten sensors has to be placed in order to maximize the coverage and the lifetime of the network. Djikstra's algorithm is repeatedly applied to the resulting topology to determine the number of rounds that can be performed provided each node has a predefined starting energy. Though the results obtained are encouraging, the small size of the network and the fact the the number of nodes is fixed instead of an optimizable value leave room for further research, as they state in their work. The NP-completeness of the WSNL problem with

heterogeneous sensor nodes (referred to as Sensor Deployment Problem by the authors) is demonstrated in [218], by assimilating it to the knapsack problem. The authors use a grid model of the terrain and propose a genetic algorithm to obtain the optimal deployment to maximize the average detection probability over the sensor field, with budget constraints on the number and types of nodes. Specific genetic crossover and mutation operators are proposed as well. A multi-objective GA is used in [105] to obtain 3D differentiated coverage by placing N sensors in a 3D field and selecting the R_{SENS} values for the nodes. Both binary and quasi unit disk coverage models are alternatively considered. The total binary coverage and the degree of differentiated coverage achieved have to be maximized, while the total energy consumption in the network (the energy consumption for a single node is considered to be proportional to R^2_{SENS}) has to be minimized. A similar problem definition, the differentiated coverage in 2D, is solved in [3] with a Tabu Search. Instead of reducing the consumed energy, the number of nodes placed has to be minimized (the R_{SENS} values can no longer be selected). The TS algorithm performs K iterations, explores a neighborhood of V solutions at each, and handles a tabu list with T elements. There are three special procedures in the TS: the initial solution generation, the neighbor generation by addition of a node, and neighbor generation by deletion of a node. In the first (initial solution), the solution is initially empty, then by a greedy procedure a node is placed at each location with probability equal to degree of violation of the coverage requirements in its coverage area, until all requirements are met. The neighbor generation by addition of a node is similar. The neighbor generation by deletion of a node deletes each node with probability equal to the degree of fulfillment of coverage requirement in that node's coverage area. The proposed TS is compared against Max_Min_Cov and Max_Avg_Aco. A GA-based memetic algorithm is proposed to solve the dynamic design of WSNs in [75]. In this problem formulation, the WSN, which operates by rounds, consists of a regular grid-deployed nodes; for each round, every node must be assigned one state out of four possibilities: cluster head, high energy operation, low energy operation, and non active. A set of objectives including active node density, energy consumption, and connectivity, are aggregated into a single weighted fitness function, and a mono-objective approach is adopted. An initial GA solution method is improved be adding a local search process that operates on a threshold basis: at each round, every node state has a corresponding remaining battery threshold; nodes that do not surpass the threshold cannot be in the corresponding state. The deployment and power assignment problem is solved using a multi-objective evolutionary algorithm in [120]. The authors propose a decomposition of the problem into several scalar problems in which the objectives, coverage (sensing disk model) and lifetime (taken as the TTFF), are merged with different weights, and reconstruct the Pareto set from the solutions to the different problems. Specific operators for mutation and crossover are proposed that operate in a different manner depending on the current objective weighting, to guide the search process towards the specific region of interest. The technique is shown to outperform NSGA-II. A GA to deploy sensors on a planar grid with obstacles and differentiated coverage is proposed in [223]. The authors adopt an indirect coding scheme, where every solution corresponds to a permutation of all the grid positions; the WSN is constructed by visiting the grid points in the specified order and adding a node in each visited point if it does not meet the coverage requirement. Since coverage levels are guaranteed (for every feasible solution), the optimization objective becomes the number of sensors. A multi-objective approach to the WSN layout, where the coverage and lifetime are the opposing objectives, and the number of nodes is fixed, is adopted in [177]; a multi-objective PSO algorithm (MOPSO) is used to solve this problem. The authors use a quasi-unit disk coverage model (they refer it as stochastic), a binary communication link model, and the TTFF criterion for the lifetime. In their problem definition, the energy spent at node level depends uniquely on the number of data packets sent by that node, and the routing is performed following Dijkstra's algorithm with link weights inversely proportional to the end node remaining energy. A MOEA, namely the IBEA, is used to solve a multi-objective sensor placement problem where the optimization objectives are the cost (measured by the number of sensor nodes) and the transmission reliability (measured by the expected transmission failure rate) in [215]. The authors employ a geographic crossover operator, and two types of mutation: a Voronoi mutation operator that either adds or removes a number of nodes according to the properties of the Voronoi graph, and a Gaussian mutation that moves nodes.

Other problems with related approaches can be found for WSNs. While not being exactly WSNL, these problems often share several issues with it (such as sought objectives). The optimal location of the BS to optimize the lifetime of a given deployed WSN is studied in [188]. The communication radius is considered unbounded ($R_{COMM} = \infty$), thus all links exist. The routing problem can be stated as a set of equations; at each node there are two (in)equations: the sum of incoming information plus the generated information equals the sum of outgoing information, and the energy consumption rate times the lifetime does not surpass the available energy; since the system can be solved, the optimal routing problem is considered as solved. The authors remark that, since nodes have fixed locations, the optimal routing strategy depends solely on the position of the BS, and the resulting transmission power required for any node to communicate with it. The proposed solution has each node define H concentric circles corresponding to its discretized levels of transmission power such that the ratio between two consecutive levels is $(1 + \epsilon)$; the intersection of all circles partitions the space into patches. The optimal routing is solved for each patch assuming the highest value of transmitted power corresponding to that path for every node, the best value is kept, and it is shown to be within ϵ of the optimum. Additionally, a modified optimal routing is proposed for the case using R_{COMM}, and an enumerative search is used to solve the case with multiple BSs. A similar problem, the optimal placement of gateways in a deployed WSN, is solved in [216]. The gateways are chosen among the nodes deployed. The optimization objectives are the latency, which is the maximum number of hops from any node to its closest gateway. The problem is defined in two ways: minimize the number of gateways for a given latency, or minimize the latency for a given number of gateways. Lower bounds are derived for the latency. The authors propose two resolution methods: an ILP formulation, and a greedy algorithm that successively eliminates candidates from the list. A complementary problem to WSNL is solved in [153]: finding the minimal exposure path. The minimal exposure path is interesting since it corresponds to the worst case scenario evaluation of the coverage. This work is later complemented in [206] by adding the maximal exposure path problem. These problems amount to finding the path between two points S and F such that the *exposure* of the path is minimal (cf. maximal). The *exposure* is defined as a value that decreases with the distance d from a node as $1/d^k$, where $2 \leq k \leq 4$ is an attenuation factor; the exposure of a path corresponds to the integration of the exposure along that path. The optimal solution for a single node is found, in polar coordinates with respect to the node, to be $\rho(\theta) = a \exp(\theta \cdot \ln(b/a)/\alpha)$, where a and b are the distances between the node and S, F, respectively. For multiple nodes, a grid approximation is used in combination with a centralized routing technique (similar to Dijkstra's). For the maximal exposure path, the path length is bounded, and four heuristic methods are proposed.

There are some big trends that can be identified when considering the resolution methodologies for WSNL. Letting aside random and regular deployments, which either do not address the problem (random deployments) or rely on a very simple problem model (regular deployments), we find two types of technique. The first type regroups specific techniques to solve a particular type of WSNL problem, also referred to as ad-hoc heuristics. In this group we have, among others, several greedy-like techniques; these techniques are very scenario-specific and thus hard to extrapolate to a different scenario, but leverage on problem knowledge and show high performances. The second type contains high-level optimization techniques, i.e., metaheuristic algorithms. These techniques are robust and versatile and can be used to solve a wide range of problem instances; however, they lack deep knowledge of the problem features that could help enhancing their performance (the use of problem-specific knowledge is restricted to just the use of special genetic operators or different fitness functions in some works). Our contribution is to propose a **combined** use of versatile **metaheuristic solvers** with a **problem-specific heuristic** to enhance their performances.

7.6 Conclusions

In this chapter we have presented and described the Wireless Sensor Network Layout optimization problem, also known as Sensor Node Deployment problem, a NP-hard problem that is widely considered as one of the most significant problems in the domain of WSNs. This problem amounts to deciding the number and

geographic positions of a set of nodes in order to produce a WSN, with the aim of maximizing the coverage and lifetime of the network, while minimizing the network cost (i.e., the number of nodes).

We have presented the most common models and assumptions adopted for coverage: at sensor node level (binary, probabilistic, and quasi-unit disk models), and at network level (point coverage, area coverage, k-coverage, perimeter coverage, differentiated, path coverage, multi-nature coverage), and the methods for the computation of the coverage (grid, Voronoi-based, disk-intersection based). For the communications, we have described the models at sensor node level (unit disk, probabilistic and quasi-unit disk), and at network level (communications model, hierarchical structure, routing models). We have discussed the lifetime and presented the most common models for its calculation. Since communications energy is assumed to be the dominant factor in the WSN energy budget, the focus when maximizing the lifetime will be on the energy consumed for communications.

Finally, we have provided an extensive literature review regarding the WSNL problem. We have noticed four principal types of approach to node deployment: random deployments, regular deployments, ad-hoc heuristic deployment methods (frequently greedy algorithms), and metaheuristic algorithms for deployment optimization. Our proposal is to combine the two latter methods: a general optimization framework (metaheuristic) enhanced by the use of a more problem-specific operator (heuristic). This will be presented in the next Chapter.

Chapter 8

Resolution Methodology and Results for Wireless Sensor Network Layout

In the previous chapter we introduced the Wireless Sensor Network Layout optimization (WSNL) as an optimization problem, described the main models used to tackle it, and reviewed the existing literature. In this chapter we address the approach adopted to solve the WSNL problem, propose a novel local improvement operator for its resolution, and discuss the results obtained.

We have adopted a constrained multi-objective approach for this problem. We have selected four state-of-the-art multi-objective optimization algorithms to solve the problem and to test the effectiveness of the proposed local improvement operator: three of them are population based techniques (NSGA-II, SPEA2 and MOCell), and the fourth is a trajectory based technique (PAES).

We initially define the formulation of WSNL, describe the models employed for the sensor node, sensor network and communications structure. We present the representations used for the solutions, and the pool of genetic operators that are required by the algorithms. Later, we present and describe the proposed operator, PACO. This operator's aim is to improve the "quality" of a candidate solution by searching for and fixing local inefficiencies in the network due to proximity of nodes. A formal definition is provided. Finally, we present the experimental setup including the different problem instances. We have defined a basic instance to perform a wide test of the operator's performance under different algorithms, different genetic operators, and different parametric configurations. In addition, we have defined two extra instances of larger size in order to test the operator's consistency for increasingly complex problem instances, and the scalability of the proposed techniques.

8.1 Problem formulation and models

The definition of the WSNL problem was already presented in Section 7.1; we expand and complete it in this section. In our formulation of the WSNL problem, coverage is treated as a constraint, with full coverage being required. The optimization objectives are then the cost of the network (which equals the number of nodes deployed and has to be minimized), and the lifetime of the network (which has to be maximized). The lifetime of the network will be defined as the **Time to First Failure**, as described in Section 7.4.

For the sensor nodes, we use **binary coverage** model, and *unit disk* model for the links, with respective radii values R_{SENS} and R_{COMM}. We are interested in **area coverage** at network level. In our formulation, a discrete **grid model** is used for the terrain, where each point in the grid represents one square meter of the terrain; the HECN is assumed to be located at the center of the terrain.

The formal definition of the problem is as follows. Let $\vec{x}$ be a vector of nodes x_i where each node is a 2D coordinate representing the node location; the length of $\vec{x}$ is non-fixed, and its nodes have to provide

full sensing coverage $C(\vec{x}) = 100$ (Eq. 8.1). The number of sensor nodes and their locations have to be chosen in a way that minimizes the cost of the network which, in this case, is calculated as the number of deployed sensor nodes (Eq. 8.2), and the energy spent in communications by the most loaded node in the network (Eq. 8.3). The load in the most loaded node of the network is minimized since this node constitutes the bottleneck of the network with respect to the network lifetime; the most loaded node will be the first node to run out of energy, hence determining the network lifetime according to the TTFF criterion (see Section 7.4). The two objectives are opposed, since the higher the number of nodes, the lower the share of retransmissions.

$$C(\vec{x}) = 100 \cdot \left(\frac{CoveredPoints(\vec{x})}{TotalPoints} \right), \tag{8.1}$$

$$Cost(\vec{x}) = Length(\vec{x}), \tag{8.2}$$

$$Energy(\vec{x}) = Max\left(\{EnergyConsumed(x_i)\}_{i=1}^{Cost(\vec{x})} \right). \tag{8.3}$$

In order to determine the energy spent in communications by any node of the WSN, the number of transmissions performed is calculated. The WSN considered operates by rounds: in a round every node collects the data from its measurements and sends it to the HECN encapsulated in a packet; between rounds the nodes are in a low-energy state. It is assumed that the main source of energy consumption is packet transmission; besides, packet (re)transmission is the sole energy-consuming process of the WSN that is directly affected by node deployment (and its resulting topology), and thus susceptible of being optimized in order to extend network lifetime. Therefore, all sources of energy consumption are neglected except packet transmissions in this work.

To calculate the energy spent by transmissions the simple wave propagation model shown in Eq. 8.4 is applied for the power required per data packet to be transmitted over from node x_i to node x_j. Assuming free-space path loss sets $\alpha = 2$. Since the β constant value does not affect the optimization problem results, it will be neglected. The total energy consumed by a node x_i is shown in Eq. 8.5, where $\beta = 1$ and $\alpha = 2$. The function $Sent(a, b)$ indicates the number of data packets sent from node a to node b (see Eq. 8.6).

$$LinkPower(x_i, x_j) = \beta \cdot ||x_i - x_j||^{\alpha}, \tag{8.4}$$

$$EnergyConsumed(x_i) = \sum_{x_j \in neighbors(x_i)} Sent(x_i, x_j) \cdot ||x_i - x_j||^2. \tag{8.5}$$

A simple load balancing routing algorithm is considered: every node sends its (re)transmitted information packets to the HECN itself if it is within communication range, or distributes them among all neighbors that are closer (in hop count) to the HECN. When there are several neighbors closer to the HECN, each of them receives a traffic share proportional to the inverse of the link power (see equations 8.4 and 8.7). Every node has a traffic (number of packets to send) equal to the packets received from nodes farther from the HECN, and additionnally produces one data packet per round (corresponding to its own sensed data) (see Eq. 8.8).

$$Sent(x_i, x_j) = Traffic(x_i) \cdot ProbSend(x_i, x_j)), \tag{8.6}$$

$$ProbSend(x_i, x_j)) = \frac{\frac{1}{||x_i - x_j||^2}}{\sum_{x_k} \frac{1}{||x_i - x_k||^2}}, \tag{8.7}$$

$$Traffic(x_i) = 1 + \sum_{x_j} Sent(x_j, x_i). \tag{8.8}$$

For this problem, a constrained multi-objective approach is adopted, by defining the objective functions f_1 and f_2, as follows:

$$f_1(\vec{x}) = Cost(\vec{x}), \tag{8.9}$$
$$f_2(\vec{x}) = Energy(\vec{x}), \tag{8.10}$$

subject to the constraint imposed be the penalty function P:

$$P(\vec{x}) = 100 - C(\vec{x}). \tag{8.11}$$

The constraint handling using a penalty function is the same that was used in the multi-objective version of the RND problem (Section 6.1).

8.2 Representation and operators

In this section we describe the representation used for the candidate solutions for the WSNL problem, and the way the are manipulated by the different genetic operators used with these solutions.

8.2.1 Solution encoding

A solution to the WSNL problem is a set of variable cardinality that contains the sensor nodes that form the network. A fixed length array of two-level genes representation is used for the solutions, similar to the representation of parameterized antennae of RND (Section 6.2.1). Each position of the solution corresponds to a potential sensor node; the first level is a binary value that marks the node as *deployed* or *undeployed*; if the network is deployed, then, the second level contains its 2D coordinate values[1]. Figure 8.1 illustrates the solution encoding used for WSNL; the length N of the solution is the maximum number of sensor nodes in the network.

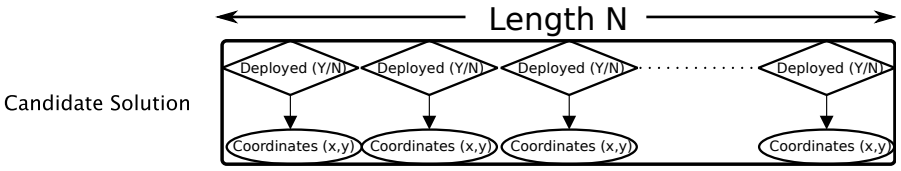


Figure 8.1: WSNL candidate solution encoding.

8.2.2 Operators

The algorithms chosen to solve WSNL are NSGA-II, SPEA2, MOCell and PAES; hence, mutation and crossover are the genetic operators that need to be defined for this problem. In WSNL, unlike RND, there is only one solution encoding. Nevertheless, there are still two possibilities selected for each of the operators: typical generic operators found in the literature (default operators), and geographic-aware operators, that

[1] Since the terrain is modeled by a point grid, the coordinates are evaluated as *integer* values. Note that this does not keep the algorithm from using real values for the coordinates anyways: when the evaluation of the solution is done, the coordinate values are rounded to the nearest integer value.

should be more problem-specific. Our intuition is that, due to the intrinsic geographic nature of the problem (with the use of geodesic coordinates), an operator that makes natural use of geographic properties should offer better performance than an operator that does not take geography into account.

Mutation operators

The mutation operators are used in all the techniques: NSGA-II, PAES, SPEA2, and MOCell. Two different mutation operators are used: a fully random mutation and a geographic mutation which is based on the polynomial mutation defined in [58]. Both mutation operators modify each potential node (which can either be deployed or not) of a given solution with some probability (the *mutation probability*, p_m); different nodes are affected by the mutation independently. When a node is chosen to be modified, the procedure differs depending on the mutation operator that is being used. Both operators first check whether the node is deployed or not. If not, they set it as deployed, and place it in a randomly generated location. Otherwise, it is either removed (set as undeployed), or repositioned with equal probability, as follows:

- Random mutation: The node is moved to any terrain point with uniformly distributed probability.
- Geographic mutation: The node is moved to a point in the surrounding area of the node's current position. This bounded movement is computed by using the polynomial mutation operator separately on the two coordinates of the node.

Figure 8.2 displays the global procedures of the two mutation operators used for WSNL.

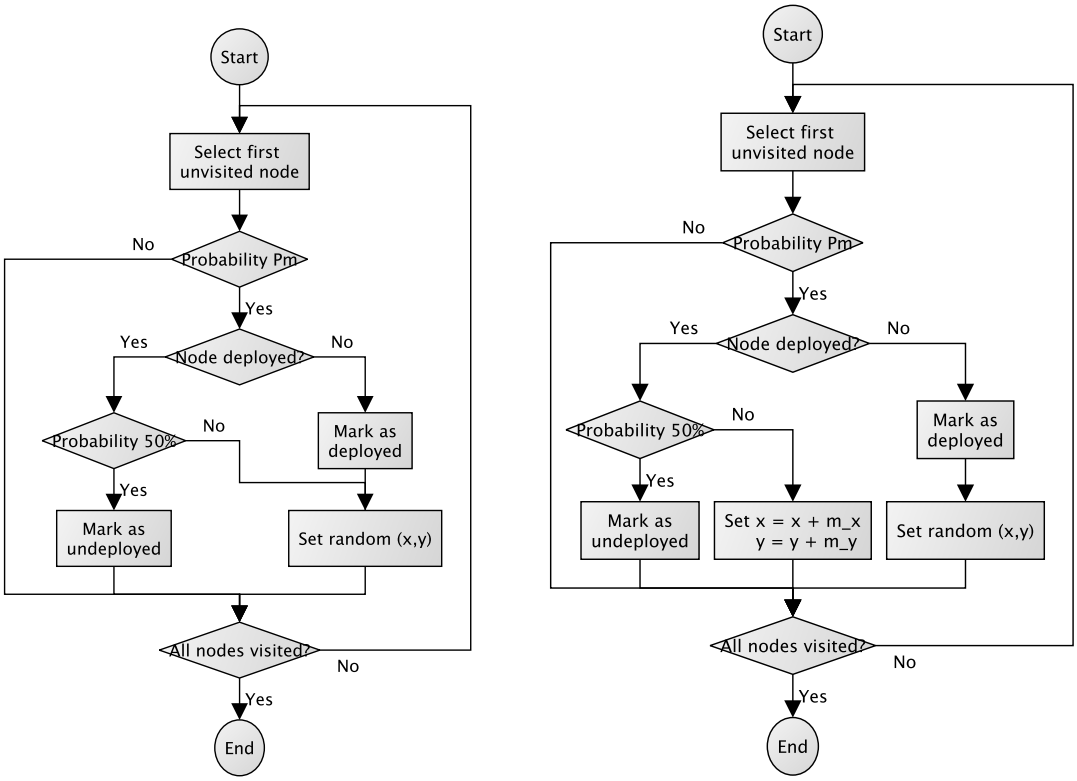
Crossover operators

The crossover operators are used in the population based techniques: NSGA-II, SPEA2, and MOCell. Two crossover operators are used: SBX ([58]) crossover and a geographic crossover ([218]). Whereas the former is the most widely applied operator in the evolutionary multi-objective community, the latter is engineered to capture the particularities of the WSN problem. In a crossover, two solutions called parents produce one or more new solutions called offspring by exchanging information with some probability (the *crossover probability*, p_c).

The main issue when adapting the SBX crossover to the solution encoding presented in Section 8.2.1 concerns the management of deployed vs. non-deployed sensors. Let p_1 and p_2 be the individuals to be crossed and let $s_i^{p_1}$ and $s_i^{p_2}$ be the sensors at position i of each individual, at which SBX is operating. Let o_1 and o_2 also be the two generated offspring and $s_i^{o_1}$ and $s_i^{o_2}$ be the corresponding sensors at the same position (i). The following cases may arise:

- Neither $s_i^{p_1}$ nor $s_i^{p_2}$ is deployed: neither $s_i^{o_1}$ nor $s_i^{o_2}$ are deployed either.
- Either $s_i^{p_1}$ or $s_i^{p_2}$ is deployed, but not both: the deployed sensor in the parent ($s_i^{p_1}$ or $s_i^{p_2}$) is independently copied to each offspring with a chance of 50%.
- Both $s_i^{p_1}$ and $s_i^{p_2}$ are deployed: the coordinates of $s_i^{o_1}$ and $s_i^{o_2}$ are computed by using the coordinates of $s_i^{p_1}$ and $s_i^{p_2}$ and the standard SBX operations. The distribution index is set to $\eta_c = 20$, a widely used value in the literature.

The other crossover operator used is the geographic crossover, called RGX (Rectangular Geographic Crossover, [218]). In it, nodes are exchanged between two solutions based on their geographic locations. A rectangular-shaped area is defined, and all nodes belonging to that area are exchanged between the two solutions (see Figure 8.3).



(a) Random (b) Geographic

Figure 8.2: Mutation operators for WSNL: (a) random mutation, (b) geographic mutation.

8.3 The PACO operator

We propose a new operator for local improvement in a WSN conceived to be integrated into an optimization algorithm: the "Proximity Avoidance Coverage-preserving Operator" (PACO). The basis of its functioning is to identify locally suboptimal configurations and try to fix them. This section presents and describes in depth the operator.

8.3.1 Operator description

We understand that, for the purpose of an efficient WSN deployment, having nodes too close to one another produces inefficiency due to two reasons:

- An extra node is deployed (increased cost) that provides little-to-no coverage improvement (since most of its sensing area is already covered by the other node).
- An extra information packet (reduced energy efficiency) containing the extra node's data has to be relayed.

Thus, the purpose of PACO is to replace the pair of nodes that are close to one another by a single node, provided that this single node can safely replace them:

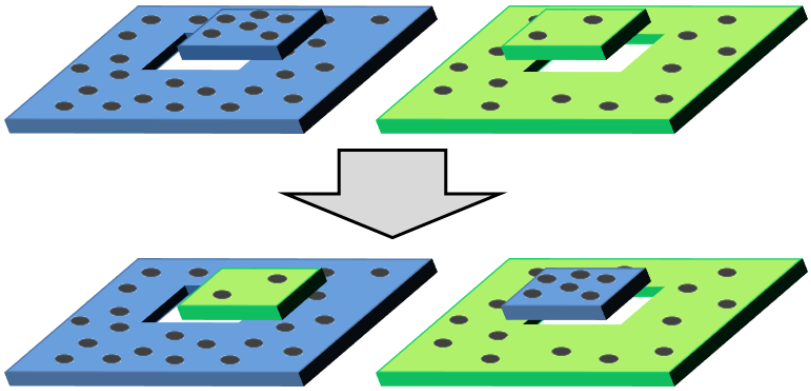


Figure 8.3: Example rectangular geographic crossover. All nodes in the extracted rectangles are exchanged between solutions.

- The node guarantees that the area covered by the two initial nodes is still covered.
- The connectivity of the WSN is maintained.

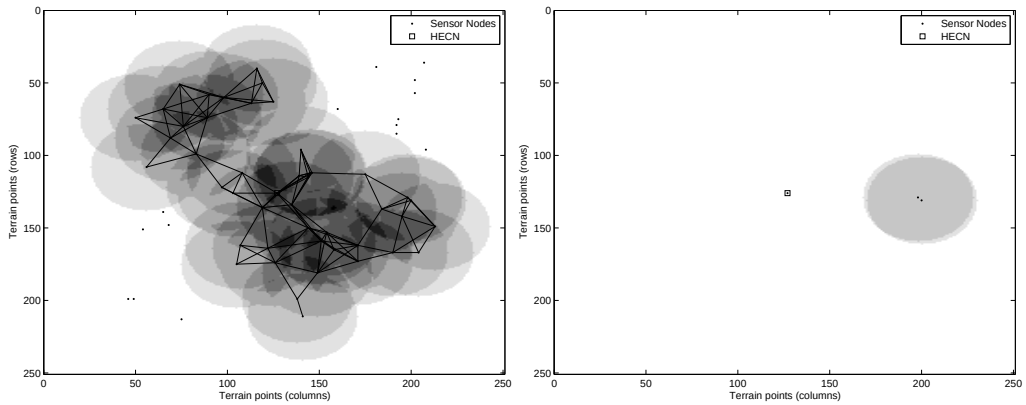
Thus PACO has to find an "equivalent deployment area" for the node pair, such that any node placed inside that area is capable of maintaining both the coverage and connectivity of the network after the pair has been removed. This area is found as the intersection of two zones: the "coverage preserving zone", which is the area where a single node guarantees coverage, and the "connectivity preserving zone", which is the area where a single node maintains the network connectivity.

It has to be pointed out that node position and covered area points are subject to a reciprocity property. If a sensor node covers a disk-shaped area around it, then any given terrain point can be covered by a sensor node placed anywhere inside that same disk-shaped area around it. This property shall be used to define a reciprocal WSN whose coverage will identify the coverage equivalent area. The same property holds for the connectivity.

The operation of PACO can be summed up in the following steps:

1. Find a pair of close nodes.
2. Obtain the "coverage preserving zone" for that pair.
3. Obtain the "connectivity preserving zone" for that pair.
4. Obtain the "equivalent deployment area" as the intersection of the two zones.
5. If the "equivalent deployment area" is not empty, replace the chosen pair of nodes by a single node in any location of the "equivalent deployment area".

The general PACO procedure is an iterative procedure (Algorithm 9). The steps above are performed for each pair of close nodes found in the WSN. We now explain them in closer detail.



(a) Example WSN

(b) Node pair selected by PACO

Figure 8.4: Example operation of PACO: Selection of close neighbors.

Selection of the pair of close nodes

The PACO operator first explores the whole WSN in search for all pairs of close nodes; this can be considered as a preliminary step. A *threshold* parameter defines which pairs of nodes are considered to be *close*: all nodes n_a, n_b, whose Euclidean distance is below it. This threshold value should typically be some fraction of R_{SENS}. In the rest of this section, the behavior of the PACO operator is illustrated with an example case: consider Figure 8.4 as the test case WSN where the operator is applied. Figure 8.4a shows the WSN with both its coverage and topology, and Figure 8.4b highlights the pair of -close- nodes that are selected by the operator, and their coverage. This network and selected pair of nodes will be the basis for all the examples below.

Coverage preserving zone

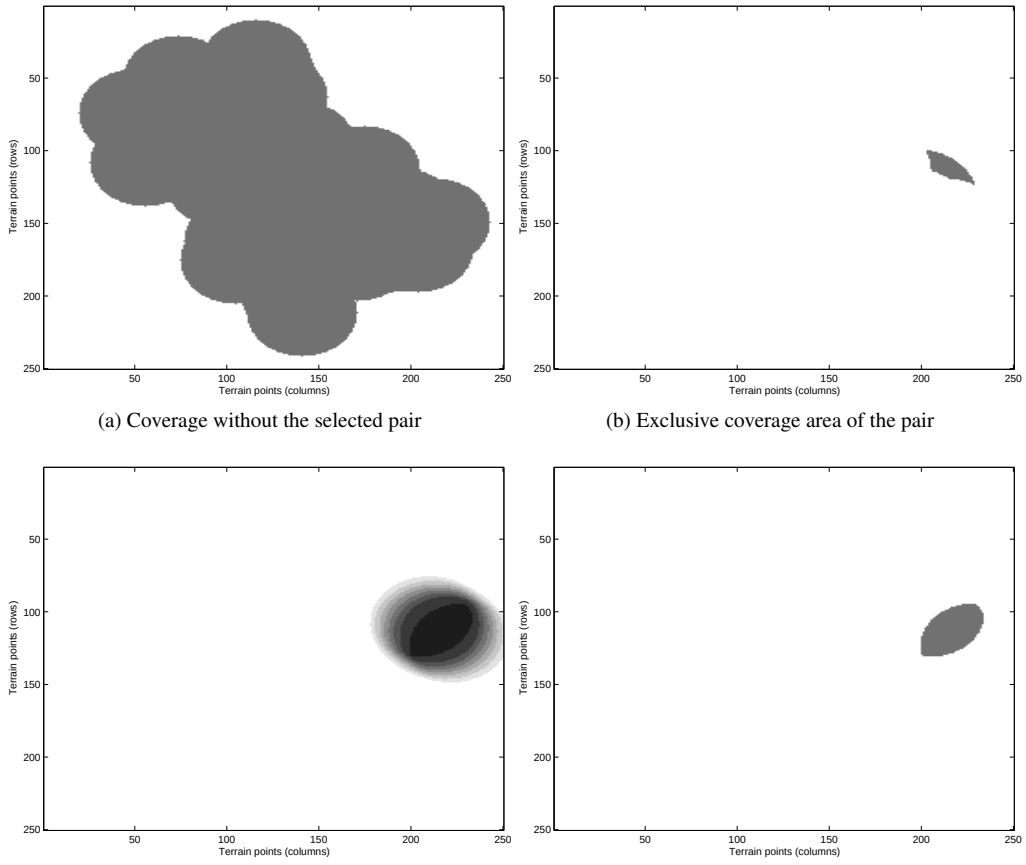
The first proper step of PACO's operation, is identifying the "coverage preserving zone". Figure 8.5 illustrates this. The coverage of the WSN is first displayed without the two chosen nodes (top left) in order to identify the area that is exclusively covered by the selected pair (top right) (note that the connectivity constraint is not taken into account here). A reciprocal WSN is then created with a node in every terrain point of that area, and the coverage of this reciprocal network is computed (bottom left); the area that is covered by all the nodes in the reciprocal WSN (bottom right) is the "coverage preserving zone". Thus, a single node placed in this zone can effectively replace the selected pair in terms of coverage.

Connectivity preserving zone

Regarding connectivity, the node has to fulfill the following constraints:

- All children nodes of the two nodes removed must be within communication range of the placed node.
- At least one of the parent nodes must be within communicating range of the placed node.

To locate the "connectivity preserving zone" the same principle as before is applied: each child and each parent defines a disk-shaped connectivity zone around itself (with radius R_{COMM}). Figure 8.6 illustrates an example case. The intersection or overlapping region (if any) of all the zones defined by the children guarantees that a single node will keep all the children connected (top right). The union of all the



(a) Coverage without the selected pair

(b) Exclusive coverage area of the pair

(c) Coverage of the reciprocal WSN

(d) Coverage preserving zone

Figure 8.5: Example operation of PACO: Coverage preserving zone.

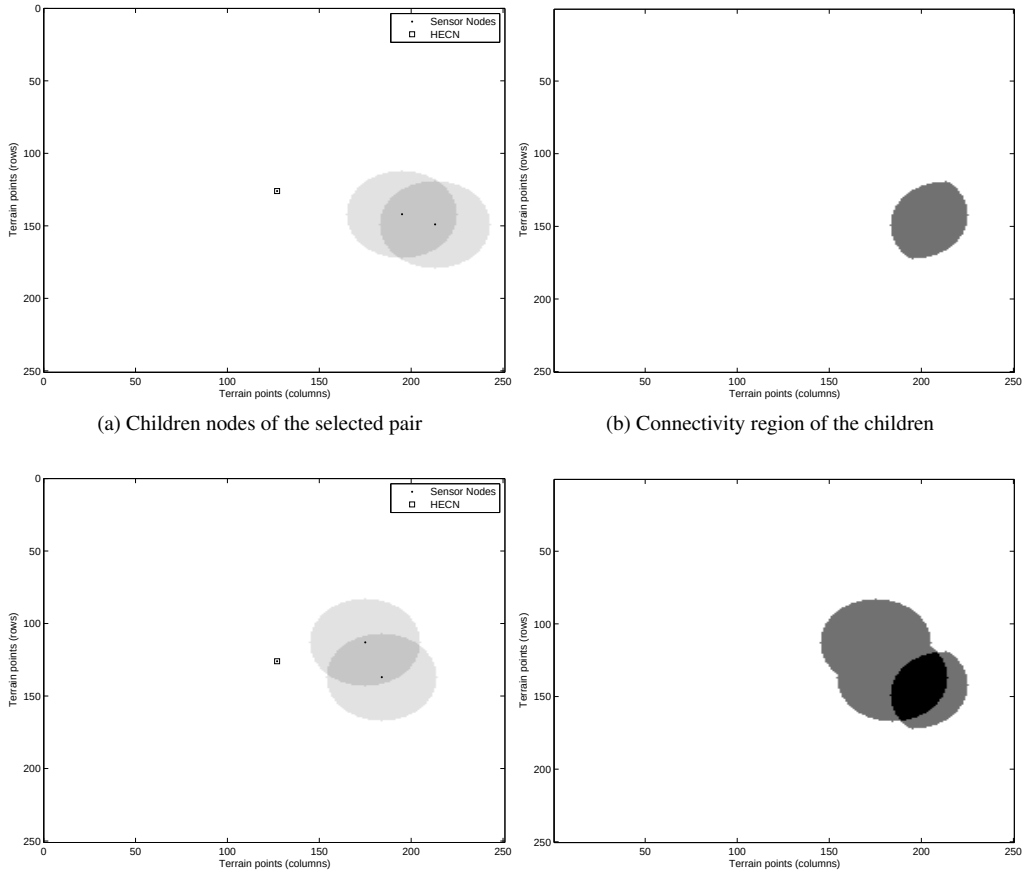
zones defined by the parent nodes guarantees that at least one parent is connected (bottom left). The final "connectivity preserving zone" is the intersection of the children and parent zones (bottom right).

Equivalent deployment area

Once both the coverage and connectivity preserving zones are determined (figures 8.5 and 8.6, respectively), the equivalent deployment area is obtained by intersecting them (see Figure 8.7). If no overlap is found between the two previous zones, the two removed nodes must be restored and the operator moves to the next pair of nodes. When there is an overlap zone (as in Figure 8.7), then a single node is placed inside it that effectively replaces the two initially chosen nodes.

8.3.2 PACO formal specification

A formal description of PACO's operation is as follows. Let T be the set of terrain points p (the discretized terrain grid), and let WSN be the points where a sensor node is deployed ($WSN \subseteq T$). Assume defined the functions $converage()$, that for each node $n \in WSN$ returns the set of points in T covered by that node, $parentNodes()$, that for each node $n \in WSN$ returns the set of nodes in WSN that are parent nodes



(a) Children nodes of the selected pair

(b) Connectivity region of the children

(c) Parent nodes of the selected pair

(d) Connectivity preserving zone

Figure 8.6: Example operation of PACO: Connectivity preserving zone.

of n, and $childNodes()$, that for each node $n \in WSN$ returns the set of nodes in WSN that are children nodes of n. Select a pair of nodes n_a and n_b such that $n_a, n_b \in WSN$ and $||n_a - n_b|| < threshold$.

- Step 1. Define E as the set of points covered only by $\{n_a, n_b\}$, i.e., $p \in E \equiv\ p \in coverage(n_a) \cup coverage(n_b);\ \forall n \in WSN, n \neq n_a, n_b,\ p \notin \cup coverage(n)$. Find the set of points $CovEq$ that guarantee coverage to the set E: $n \in CovEq \equiv \forall p \in T : p \in E \rightarrow p \in coverage(n)$.

- Step 2. Define the sets P and C such that: $P = parentNodes(n_a) \cup parentNodes(n_b)$ and $C = childNodes(n_a) \cup childNodes(n_b)$. Then find the set $ConEq$ that maintains the connectivity of the network: $n \in ConEq \equiv \forall n_c \in WSN : n_c \in C \rightarrow n_c \in childNodes(n)$, $\exists n_p \in P : n_p \in parentNodes(n)$.

- Step 3. Define $CovConEq$ as the set of points that guarantee both coverage and connectivity: $CovConEq = CovEq \cap ConEq$.

Then, as long as $CovConEq \neq \varnothing$, a single sensor placed in any $n \in CovConEq$ may replace the pair $\{n_a, n_b\}$ without loss of coverage or connectivity.

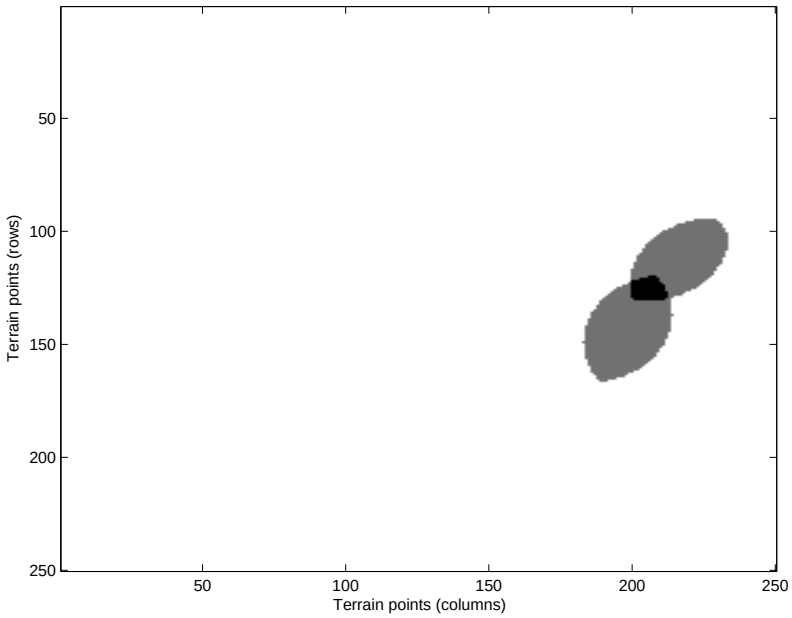


Figure 8.7: Example operation of PACO: Equivalent deployment area obtained by intersection of coverage and connectivity preserving zones.

Algorithm 9 Pseudocode for PACO.

input: a WSN layout $wsn = n_1 n_2 \dots n_k$, $n_i \in WSN$, a threshold value th
$wsnBackup \leftarrow wsn$ // Store a copy of the current layout
stop $\leftarrow$ false
for All (n_a, n_b) $\leftarrow$ NodePair(wsn) **do**
  if NearbyNodes(n_a,n_b, th) **then**
    $CovEq \leftarrow$ ComputeCovEq(wsn,n_a,n_b) // Step 1
    $ConEq \leftarrow$ ComputeConEq(wsn,n_a,n_b) // Step 2
    $CovConEq \leftarrow CovEq \cap ConEq$ // Step 3
    if $CovConEq \neq \emptyset$ **then**
      $n_p \leftarrow$ ChooseNode(wsn,$CovConEq$)
      $wsn \leftarrow$ Remove(wsn,n_a,n_b)
      $wsn \leftarrow$ Deploy(wsn,n_p)
      Evaluate(wsn)
    end if
  end if
end for
if NodesDeployed(wsn) $<$ NodesDeployed($wsnBackup$) & EnergyConsumption(wsn) $<$ EnergyConsumption($wsnBackup$) **then**
  return wsn // wsn dominates $wsnBackup$
else
  return $wsnBackup$
end if
output: a possibly improved WSN layout

8.4 Problem instances

We define a **basic instance** for the WSNL problem as follows:

- Terrain: square grid of $250 \times 250m^2$.
- HECN located at the center of the terrain.
- Maximum number of sensor nodes: 250.
- Initial node deployment probability 50%, uniform distribution.
- $R_{SENS} = 30m$.
- $R_{COMM} = 30m$.
- Wave propagation model: $P = d^2$, where P is the required power to send, d is the distance traveled by the signal.
- We neglect the energy consumption associated with sensing, processing and signal reception.

For this instance, full coverage is required (100%).

8.5 Experiments

We present in this section the results obtained in the experimental study conducted for the WSNL problem. We have used four multi-objective algorithms as the test techniques to solve the problem: NSGA-II, PAES, SPEA2, and MOCell (their descriptions can be found in Chapter 4). The parametric configurations, obtained empirically, are shown in Table 8.1.

Table 8.1: Parametric configuration of the optimization algorithms used in WSNL.

Algorithm	**NSGA-II**
population	100
selection	binary tournament
crossover	SBX geographic
mutation	random geographic
replacement	ranking and crowding

Algorithm	**PAES**
archive	100
mutation	random geographic
replacement	ranking and crowding

Algorithm	**SPEA2**
population	100
selection	binary tournament
crossover	SBX geographic
mutation	random geographic
replacement	ranking system

Algorithm	**MOCell**
population	100
selection	binary tournament in cellular neighborhood
crossover	SBX geographic
mutation	random geographic
replacement	ranking and crowding in cellular neighborhood

The internal parameters of the PACO operator were also empirically tuned, with the resulting configuration obtained as the best performing:

- Probability of use: 100%.
- Threshold: $30m$.

For each problem instance and algorithm, the stopping condition of an execution is $1,000,000$ solution evaluations.

8.5.1 Results for the basic instance

The basic instance defined above serves as the test bench to assess the effectiveness of the PACO improvement operator in various scenarios. Therefore, we test the four algorithms with different genetic operators and parametric configurations, each of which both using PACO, and not using it. With this, we will be able to assess the effectiveness of the operator, its robustness facing different optimization techniques, and the degree of improvement that can be expected by using it.

For commodity, the results of all the executions corresponding to the basic problem instance are displayed in Table 8.2. We show the median and inter-quartile range of the hypervolume indicator obtained in the 30 executions performed for each algorithm.

Effectiveness of the PACO operator

The HV values displayed in Table 8.2 vary from 0.0 to 0.768 (they are normalized to unity). The configurations integrating PACO produce higher HV than the same configurations without PACO in 112 of 132 test configurations, that is, PACO produces an improved efficiency in 84.85% of the cases. However, some of these test configurations produce poor performances in either case, thus their results are not very meaningful. If we restrict the comparison to the high-performing configurations (the best half), then PACO yields improved performance in 98.48% of the cases. Therefore, PACO is a robust technique, and is best performing when used in combination with a high performing algorithmic configuration.

Comparison of the genetic operators and parametric configurations

Regarding the mutation operator, polynomial mutation clearly outperforms random mutation: in all of the 132 test configurations, the HV obtained with polynomial mutation is always higher than that of random mutation (100% improved efficiency). For the crossover operator, RGX produces the best results: in the 108 test configurations (we exclude the ones where the crossover is not involved), RGX always obtained higher HV than SBX (again 100% improved efficiency). Furthermore, for the three algorithms including crossover (NSGA-II, SPEA2 and MOCell), the best configuration with crossover outperforms 100% of the time the one without crossover (i.e., with $p_c = 0$).

Regarding the parametric configuration, the dominant factor seems to be the mutation probability, with the highest HV values obtained for $p_m = 1.0$. For the crossover, the probability does not have such a big influence, but the best results are generally obtained with $p_c = 0.5$.

Due to the high dimensionality of the scenarios considered, we carry the statistical analysis only among the best parametric configurations. The statistical analysis results prove that the algorithmic configurations "SBX + randomMutation + PACO/noPACO" and "RGX + randomMutation + noPACO" are statistically similar, while being statistically worse than the rest. On the other side, the configuration "RGX + polynomialMutation + PACO" is statistically better than the rest.

Comparison of the algorithms

Finally, when we compare the algorithms, the results are less evident. MOCell obtains the highest HV values (the 10 best performing parametric configurations obtain their highest HV values with MOCell), but is quite sensitive to the operator configuration and is outperformed by NSGA-II in the big picture (of the

Table 8.2: Performance of PACO with different genetic operators: median and IQR of the HV indicator.

Crossover operator			SBX				RGX			
Mutation operator			Polynomial		Random		Polynomial		Random	
Algorithm	p_m	p_c	no PACO	**PACO**	no PACO	**PACO**	no PACO	**PACO**	no PACO	**PACO**
NSGAII	1.0	0.0	$0.629_{0.043}$	$0.679_{0.033}$	$0.489_{0.045}$	$0.593_{0.057}$	$0.629_{0.043}$	$0.679_{0.033}$	$0.489_{0.045}$	$0.593_{0.057}$
		0.1	$0.640_{0.041}$	$0.672_{0.039}$	$0.498_{0.045}$	$0.590_{0.046}$	$0.698_{0.040}$	$0.730_{0.052}$	$0.550_{0.071}$	$0.643_{0.049}$
		0.5	$0.636_{0.050}$	$0.649_{0.042}$	$0.501_{0.054}$	$0.573_{0.052}$	$0.677_{0.041}$	$0.731_{0.051}$	$0.520_{0.068}$	$0.634_{0.060}$
		0.9	$0.606_{0.059}$	$0.603_{0.067}$	$0.478_{0.065}$	$0.551_{0.094}$	$0.671_{0.035}$	$0.720_{0.050}$	$0.523_{0.030}$	$0.646_{0.070}$
	5.0	0.0	$0.508_{0.029}$	$0.533_{0.034}$	$0.463_{0.036}$	$0.485_{0.042}$	$0.508_{0.029}$	$0.533_{0.034}$	$0.463_{0.036}$	$0.485_{0.042}$
		0.1	$0.504_{0.045}$	$0.538_{0.032}$	$0.448_{0.040}$	$0.481_{0.045}$	$0.528_{0.059}$	$0.565_{0.043}$	$0.478_{0.047}$	$0.513_{0.018}$
		0.5	$0.456_{0.042}$	$0.462_{0.041}$	$0.398_{0.055}$	$0.424_{0.043}$	$0.565_{0.039}$	$0.578_{0.048}$	$0.532_{0.058}$	$0.551_{0.060}$
		0.9	$0.339_{0.059}$	$0.295_{0.092}$	$0.289_{0.072}$	$0.262_{0.055}$	$0.570_{0.052}$	$0.608_{0.050}$	$0.528_{0.073}$	$0.560_{0.060}$
	10.0	0.0	$0.149_{0.036}$	$0.151_{0.033}$	$0.087_{0.027}$	$0.115_{0.037}$	$0.149_{0.036}$	$0.151_{0.033}$	$0.087_{0.027}$	$0.115_{0.037}$
		0.1	$0.123_{0.032}$	$0.127_{0.028}$	$0.088_{0.036}$	$0.097_{0.034}$	$0.152_{0.032}$	$0.152_{0.031}$	$0.104_{0.037}$	$0.124_{0.027}$
		0.5	$0.084_{0.035}$	$0.078_{0.039}$	$0.048_{0.030}$	$0.048_{0.031}$	$0.171_{0.045}$	$0.183_{0.040}$	$0.134_{0.042}$	$0.153_{0.049}$
		0.9	$0.008_{0.017}$	$0.003_{0.014}$	$0.001_{0.010}$	$0.001_{0.005}$	$0.201_{0.053}$	$0.207_{0.045}$	$0.171_{0.053}$	$0.158_{0.054}$
SPEA2	1.0	0.0	$0.569_{0.060}$	$0.625_{0.049}$	$0.450_{0.036}$	$0.538_{0.058}$	$0.569_{0.060}$	$0.625_{0.049}$	$0.450_{0.036}$	$0.538_{0.058}$
		0.1	$0.569_{0.043}$	$0.633_{0.060}$	$0.475_{0.060}$	$0.536_{0.057}$	$0.608_{0.048}$	$0.684_{0.055}$	$0.510_{0.064}$	$0.599_{0.038}$
		0.5	$0.579_{0.061}$	$0.639_{0.054}$	$0.459_{0.063}$	$0.518_{0.068}$	$0.626_{0.036}$	$0.662_{0.033}$	$0.489_{0.068}$	$0.584_{0.045}$
		0.9	$0.559_{0.070}$	$0.598_{0.054}$	$0.450_{0.063}$	$0.520_{0.060}$	$0.610_{0.044}$	$0.666_{0.046}$	$0.490_{0.055}$	$0.583_{0.048}$
	5.0	0.0	$0.490_{0.028}$	$0.514_{0.048}$	$0.447_{0.049}$	$0.483_{0.043}$	$0.490_{0.028}$	$0.514_{0.048}$	$0.447_{0.049}$	$0.483_{0.043}$
		0.1	$0.487_{0.032}$	$0.516_{0.051}$	$0.436_{0.032}$	$0.465_{0.042}$	$0.515_{0.037}$	$0.540_{0.039}$	$0.463_{0.048}$	$0.488_{0.045}$
		0.5	$0.447_{0.033}$	$0.463_{0.048}$	$0.404_{0.060}$	$0.413_{0.053}$	$0.539_{0.044}$	$0.576_{0.060}$	$0.502_{0.044}$	$0.545_{0.063}$
		0.9	$0.353_{0.068}$	$0.299_{0.060}$	$0.313_{0.077}$	$0.266_{0.100}$	$0.564_{0.052}$	$0.588_{0.060}$	$0.518_{0.037}$	$0.542_{0.055}$
	10.0	0.0	$0.137_{0.029}$	$0.157_{0.019}$	$0.094_{0.034}$	$0.126_{0.036}$	$0.137_{0.029}$	$0.157_{0.019}$	$0.094_{0.034}$	$0.126_{0.036}$
		0.1	$0.119_{0.041}$	$0.137_{0.023}$	$0.087_{0.032}$	$0.111_{0.025}$	$0.147_{0.037}$	$0.155_{0.029}$	$0.111_{0.033}$	$0.119_{0.031}$
		0.5	$0.071_{0.028}$	$0.068_{0.029}$	$0.047_{0.021}$	$0.048_{0.023}$	$0.171_{0.035}$	$0.180_{0.026}$	$0.128_{0.031}$	$0.157_{0.048}$
		0.9	$0.004_{0.009}$	$0.001_{0.005}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.207_{0.054}$	$0.209_{0.061}$	$0.173_{0.053}$	$0.177_{0.038}$
PAES	1.00	N/A	$0.568_{0.086}$	$0.645_{0.078}$	$0.470_{0.107}$	$0.542_{0.041}$	$0.568_{0.086}$	$0.645_{0.078}$	$0.470_{0.107}$	$0.542_{0.041}$
	5.0	N/A	$0.535_{0.060}$	$0.562_{0.057}$	$0.445_{0.062}$	$0.492_{0.064}$	$0.535_{0.060}$	$0.562_{0.057}$	$0.445_{0.062}$	$0.492_{0.064}$
	10.0	N/A	$0.229_{0.041}$	$0.256_{0.036}$	$0.188_{0.056}$	$0.218_{0.056}$	$0.229_{0.041}$	$0.256_{0.036}$	$0.188_{0.056}$	$0.218_{0.056}$
MOCell	1.0	0.0	$0.648_{0.028}$	$0.685_{0.029}$	$0.492_{0.043}$	$0.600_{0.058}$	$0.648_{0.028}$	$0.685_{0.029}$	$0.492_{0.043}$	$0.600_{0.058}$
		0.1	$0.636_{0.034}$	$0.683_{0.038}$	$0.508_{0.050}$	$0.587_{0.060}$	$0.728_{0.047}$	$0.760_{0.032}$	$0.605_{0.079}$	$0.682_{0.047}$
		0.5	$0.603_{0.054}$	$0.647_{0.057}$	$0.478_{0.083}$	$0.550_{0.056}$	$0.724_{0.043}$	$0.768_{0.042}$	$0.566_{0.059}$	$0.674_{0.061}$
		0.9	$0.559_{0.056}$	$0.595_{0.060}$	$0.428_{0.087}$	$0.530_{0.080}$	$0.700_{0.048}$	$0.749_{0.058}$	$0.547_{0.051}$	$0.650_{0.045}$
	5.0	0.0	$0.448_{0.048}$	$0.469_{0.043}$	$0.375_{0.036}$	$0.402_{0.052}$	$0.448_{0.048}$	$0.469_{0.043}$	$0.375_{0.036}$	$0.402_{0.052}$
		0.1	$0.423_{0.045}$	$0.440_{0.039}$	$0.368_{0.051}$	$0.397_{0.062}$	$0.441_{0.042}$	$0.483_{0.047}$	$0.402_{0.066}$	$0.442_{0.060}$
		0.5	$0.372_{0.071}$	$0.370_{0.063}$	$0.337_{0.036}$	$0.348_{0.039}$	$0.516_{0.042}$	$0.525_{0.047}$	$0.501_{0.040}$	$0.514_{0.055}$
		0.9	$0.374_{0.055}$	$0.385_{0.085}$	$0.351_{0.059}$	$0.304_{0.094}$	$0.566_{0.046}$	$0.575_{0.035}$	$0.523_{0.051}$	$0.540_{0.038}$
	10.0	0.0	$0.079_{0.021}$	$0.100_{0.039}$	$0.056_{0.037}$	$0.064_{0.028}$	$0.079_{0.021}$	$0.100_{0.039}$	$0.056_{0.037}$	$0.064_{0.028}$
		0.1	$0.077_{0.038}$	$0.091_{0.041}$	$0.046_{0.030}$	$0.063_{0.026}$	$0.095_{0.032}$	$0.105_{0.045}$	$0.054_{0.026}$	$0.070_{0.031}$
		0.5	$0.040_{0.029}$	$0.031_{0.024}$	$0.019_{0.018}$	$0.030_{0.027}$	$0.115_{0.041}$	$0.126_{0.047}$	$0.081_{0.030}$	$0.101_{0.024}$
		0.9	$0.021_{0.023}$	$0.011_{0.017}$	$0.000_{0.007}$	$0.001_{0.007}$	$0.159_{0.049}$	$0.154_{0.035}$	$0.127_{0.024}$	$0.126_{0.032}$

84 test configurations, NSGA-II outperforms MOCell in 59, hence in 70.24%). SPEA2 and PAES produce lower HV values than NSGA-II or MOCell. In the statistical tests, the best configuration of MOCell is the one that most often outperforms any other configuration; as a matter of fact, for any combination of mutation, crossover and PACO operator, MOCell with $p_m = 1.0$ and $p_c = 0.5$ systematically obtains the highest number of wins against other algorithms and/or configurations, or is at least tied for highest number of wins.

Expected front improvement with PACO

In order to show deeper insight on the improvement that can be expected by using PACO within an optimization algorithm, we display the 50%-attainment surfaces obtained by the best configuration of each algorithm both with PACO and without PACO in Figure 8.8. As we can see in figures 8.8b and 8.8c, for both SPEA2 and PAES the attainment surfaces of PACO completely dominate the ones without PACO. For NSGA-II (Figure 8.8a) the region where the number of nodes is below 70 is noticeably dominated by PACO, whereas the one where number of nodes is $n > 70$ seems indistinguishable between PACO and no PACO. Finally, for MOCell (Figure 8.8d) the attainment surface of PACO when $n < 80$ clearly dominates the one without PACO, when $80 < n < 88$ both attainment surfaces are equivalent, and when $n > 88$ the configuration with PACO does not find any point and is dominated by the one without PACO; nevertheless, this last region contains only 3 points while the region $n < 80$ contains 20, and besides the energy consumption gain is only marginal, therefore the configuration using PACO globally outperforms the one without PACO.

From the problem's perspective, we can say that for any given number of nodes, the algorithmic configuration using PACO can find a solution that achieves full terrain coverage with lower energy consumption than the same algorithm without PACO, and the differences become more clear as the number of nodes is reduced.

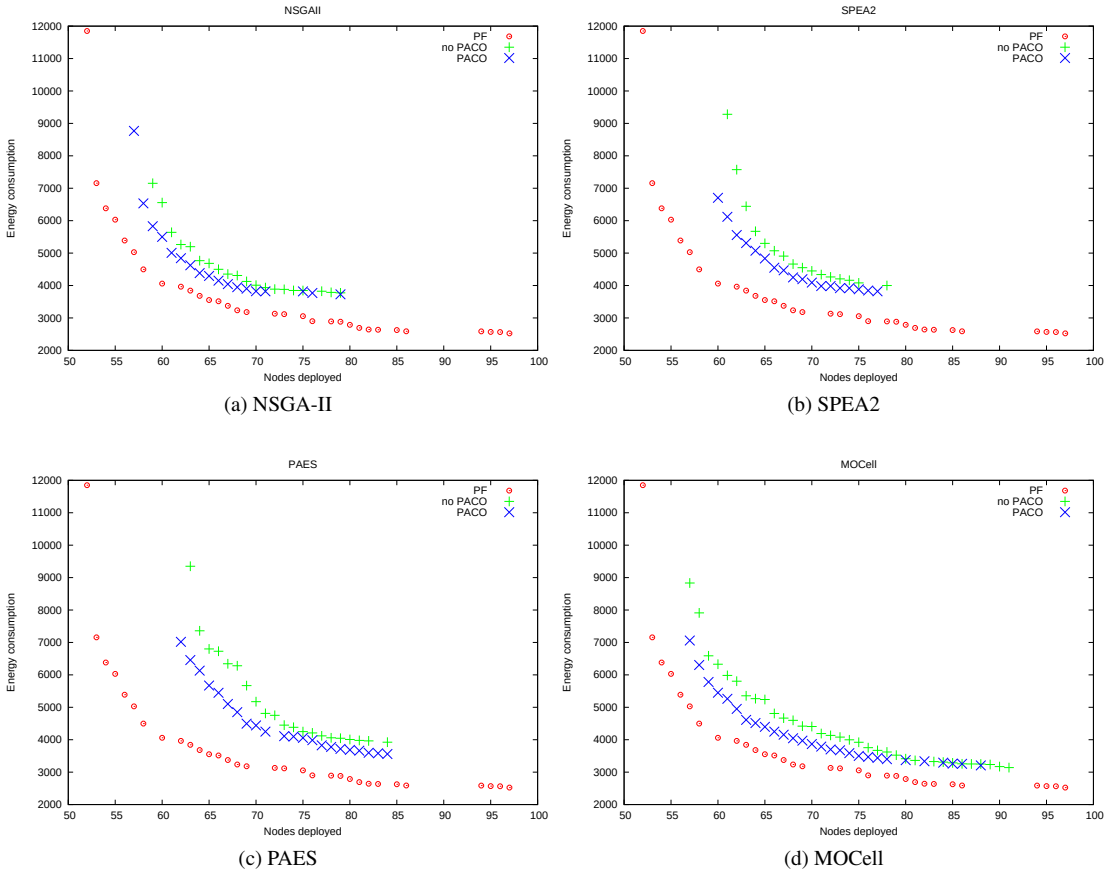
8.5.2 Sensibility to node density

Our next step is to test the sensibility of PACO towards the node density in the WSN. For this, we modify the initial node deployment probability X. Besides the predefined probability of $X = 50\%$ (for the basic problem instance), we test the values $X = 75\%$ and $X = 100\%$. The experiment is performed with the four optimization algorithms with a standard parametric configuration, stopping after evaluating $1,000,000$ solutions. The results of this experiment (HV median and IQR) are shown in Table 8.3.

Table 8.3: Influence of the initial conditions on PACO: HV. Median and IQR

X	NSGAII		SPEA2		PAES		MOCell	
	no PACO	**PACO**	no PACO	**PACO**	no PACO	**PACO**	no PACO	**PACO**
50%	$0.606_{0.059}$	$0.628_{0.068}$	$0.559_{0.070}$	$0.598_{0.060}$	$0.568_{0.086}$	$0.655_{0.057}$	$0.559_{0.056}$	$0.593_{0.040}$
75%	$0.608_{0.044}$	$0.616_{0.059}$	$0.557_{0.043}$	$0.596_{0.050}$	$0.568_{0.052}$	$0.610_{0.042}$	$0.544_{0.057}$	$0.610_{0.065}$
100%	$0.593_{0.065}$	$0.613_{0.058}$	$0.538_{0.039}$	$0.600_{0.031}$	$0.554_{0.065}$	$0.605_{0.065}$	$0.551_{0.059}$	$0.601_{0.045}$

Again the results show clearly the benefits of using PACO: in the twelve scenarios consisting of combining algorithm and starting node density, the configuration using PACO outperforms the one without it. Furthermore, with the exception of PAES, the results obtained with PACO-equipped algorithms are fairly stable over the range of initial node densities, while the configurations without PACO always experience clear performance degradations, therefore demonstrating the robustness of the operator for varying node densities.



(a) NSGA-II (b) SPEA2

(c) PAES (d) MOCell

Figure 8.8: 50%-attainment surfaces of the optimization algorithms with and without PACO. The global non-dominated fronts are represented for comparison, labeled as 'PF'.

8.5.3 Scalability study

In this section, we conduct a set of experiments to explore the scalability properties of the PACO operator, when the instances solved have increasing dimension. For this, in addition to the basic problem instance (defined in Section 8.4), we define the following two problem instances:

- Square terrain: $500 \times 500\, m^2$, maximum number of nodes: $1,000$.
- Square terrain: $750 \times 750\, m^2$, maximum number of nodes: $2,000$.

The rest of features (corresponding to the node and communications models) are left unchanged. These two instances shall be named "instance 500" and "instance 750" for brevity.

We run the four optimization techniques under their best configurations found for the basic problem instance (polynomial mutation with $p_m = 1.0$, RGX crossover with $p_c = 0.5$) to solve the two newly defined instances. For each scenario, both the version with PACO and the version without PACO are run. Table 8.4 shows the HV median and interquartile values obtained for the set of experiments.

Our first remark is that PACO produces greater gains in performance for larger instances: in the instance 250, the HV value increases by less than 10%, in the instance 500 by at least 29%, and in the instance 750

Table 8.4: Scalability properties of the different algorithmic instances (HV. Median and IQR)

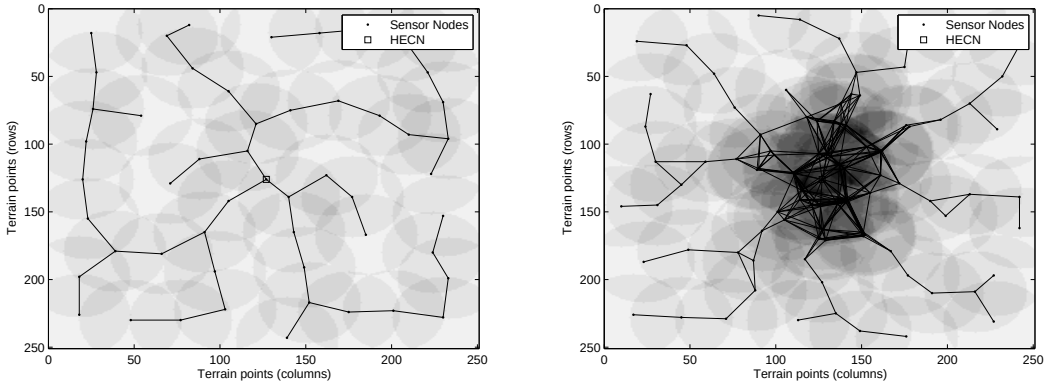
Instance	NSGAII		SPEA2		PAES		MOCell	
	no PACO	**PACO**	no PACO	**PACO**	no PACO	**PACO**	no PACO	**PACO**
250	$0.677_{0.041}$	$0.731_{0.051}$	$0.626_{0.036}$	$0.662_{0.033}$	$0.535_{0.060}$	$0.562_{0.057}$	$0.724_{0.043}$	$0.768_{0.042}$
500	$0.392_{0.063}$	$0.532_{0.114}$	$0.287_{0.066}$	$0.465_{0.070}$	$0.380_{0.100}$	$0.526_{0.108}$	$0.483_{0.088}$	$0.624_{0.065}$
750	$0.000_{0.000}$	$0.232_{0.131}$	$0.000_{0.000}$	$0.105_{0.147}$	$0.288_{0.163}$	$0.602_{0.079}$	$0.023_{0.060}$	$0.431_{0.140}$

by more than 109%. Regarding the algorithms, the population-based techniques (NSGA-II, SPEA2, and MOCell) suffer HV degradation when the instance grows, MOCell still outperforming the other two. PAES, on the other side, seems unaffected; in the instance 750, PAES is the best performing technique. There are two reasons: first, as the instance grows so does the spreadth of the nondominated front, hence a larger population should be used; second, the difference between solutions in the front is also larger, which can make the crossover operator too disruptive (crossing two very different solutions will not likely improve either of them).

8.5.4 Solutions obtained for the WSNL problem

We have relied so far on quality estimators as the tools to establish comparisons among different techniques and configurations, and to assess the effectiveness of the PACO operator. However useful these tools are for the aforementioned purposes, they lack to provide insight on what solutions are actually being obtained for the problems at hand. Therefore, we will briefly discuss the solutions obtained for the WSNL problem using the optimization techniques with the PACO operator in this section.

Figure 8.9 shows the solutions at the two opposite ends of the non-dominated front generated by MOCell equipped with PACO: Figure 8.9a plots the solution with the minimum number of nodes, and Figure 8.9b plots the solution with larger lifetime.



(a) Minimum of number of nodes

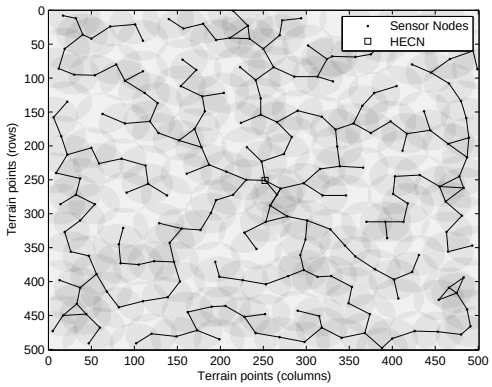
(b) Minimum of consumed energy

Figure 8.9: Best performing solutions produced by MOCell using PACO for the basic instance: $250 \times 250\,m^2$

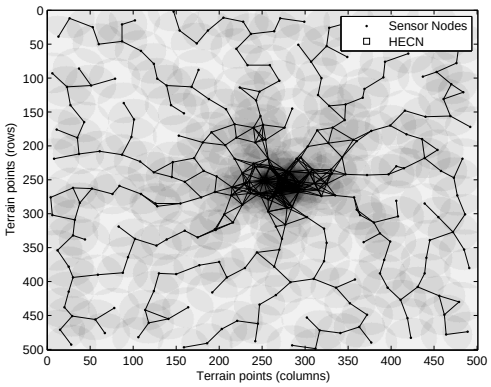
We can observe that in the solution displayed in Figure 8.9a, the network is sparse, with the nodes being deployed with a seemingly homogeneous density throughout the terrain field; this is expected, since if one tries to minimize the number of nodes to produce complete coverage, the nodes will be as far away from one another as possible. Meanwhile, in the solution displayed in Figure 8.9b, there is a high node density in the proximity of the HECN (towards the geographic center of the terrain), while the nodes at

the periphery have a much lower density. This also matches intuition, since according to our definition of lifetime (TTFF), the lifetime value is determined by the bottleneck (i.e., the node that consumes the most energy of the network), while the energy is determined by the total number of transmitted messages per round, and by the energy consumed per transmitted message (link distance). Since nodes in the proximity of the HECN have to retransmit the messages from all the other nodes of the network to the HECN itself, they are typically the bottlenecks. By increasing their number one can reduce their share of retransmitted messages, and by reducing their distance to the HECN one reduces the energy consumed per transmitted message; both mechanisms prolong the lifetime, but increase the node density in the proximity of the HECN at the same time.

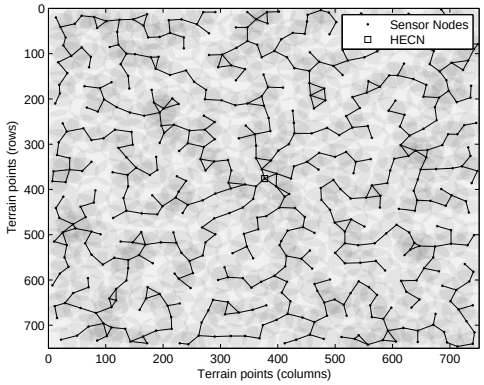
In a similar manner, Figure 8.10 displays the extremal solutions obtained for the larger-sized instances, 500 and 750. Figures 8.10a and 8.10c show the respective solutions with minimum number of nodes, while figures 8.10b and 8.10d show the respective solutions with optimal network lifetime.



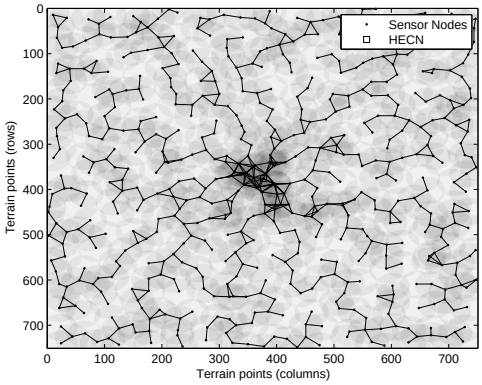
(a) 500: Minimum of number of nodes



(b) 500: Minimum of consumed energy



(c) 750: Minimum of number of nodes



(d) 750: Minimum of consumed energy

Figure 8.10: Best performing solutions produced by MOCell using PACO for the larger instances: $500 \times 500\,m^2$ and $750 \times 750\,m^2$

As in the case for the basic instance, the solutions with minimum number of sensor nodes have a low constant node density, while the solutions with optimal lifetime have a low density of nodes in the peripheral region and an accumulation of nodes (high density) in the center of the terrain, around the HECN. As said

before, this accumulation of nodes in the center alleviated the energy consumption of each of these nodes and increases the network lifetime.

However, unlike what happens for the basic instance, when the instance dimension grows the observed difference between the two extremal solutions is apparently reduced. This can also be explained, however. As the dimension of the problem increases, so does the solution space, and at the same time the number of possible points in the non dominated front. Since the algorithm configurations are tailored to the basic problem instance, they are insufficient for larger search spaces and do not manage to cover but a small fraction of the front, corresponding to the zone with lower node density.

8.6 Conclusions

In this chapter we have presented the formulation adopted for the WSNL problem, and the resolution process. Our formulation of WSNL defines it as a multi-objective problem where the number of nodes and the energy consumed in communications are considered optimization objectives, while the coverage is treated as a constraint, with full coverage being required.

We propose a novel local improvement operator, PACO, to be integrated within an optimization algorithm, to tackle the WSNL problem. The PACO operator works by finding pairs of close nodes in the network that may constitute a source of inefficiency, and searches for a single node that might replace them while improving the solution quality; PACO is used by the optimization algorithms at the end step of each iteration. We describe the operator in depth, and provide a formulation for it.

The effectiveness of the PACO operator is assessed by integrating it into four different state-of-the-art multi-objective optimization algorithms: NSGA-II, PAES, SPEA2, and MOCell. Our test bench is a basic instance where the performances of the canonical algorithms with different genetic operators and parametric configurations are compared against those of the same configurations equipped with PACO; the results show without doubt that the algorithms equipped with PACO achieve higher HV values, and that the expected performance improvements are higher when the starting algorithmic configuration is already a high-performing one.

Additionally, we test the sensitivity of PACO towards the initial density of nodes in the terrain, and its scalability when the size of the terrain and the number of nodes are increased. In the first experiment, PACO proved to be robust against variations in the node density, while in the second, it proved to be scalable, since the algorithmic configurations equipped with PACO consistently obtained increasingly higher HV values than the same configurations without PACO.

Part IV

LOCATION DISCOVERY

Chapter 9

Location Discovery in Wireless Sensor Networks

The location information is a fundamental aspect for the functioning of WSNs, without which many applications of these networks could not be made possible. In fact, many of the basic mechanisms used during normal operation of WSNs require the use of location information; that is, sensor nodes need to have their location information for the WSN to work properly. For instance, in data aggregation, or data integration, the operation is performed on a location-basis, i.e., the relationship among sensed data depends on the geographic distance between sensing nodes. For data-centric operation purposes, location information is also important: nodes in a WSN are not addressed in a computer-like manner (with an IP address); instead they are addressed according to their properties, among which is their location (for instance, one may want to check all nodes of a given region to see what is happening there). Other processes that make use of the location information are many distributed geographic-based routing techniques, like Geographic Forwarding or Greedy Perimeter Stateless Routing. In self-evaluation techniques such as distributed coverage evaluation, the nodes need to know their location in order to estimate their and the network's coverage. Many times it is the very main application of the WSN that intrinsically needs location information: in a target location and tracking application, neither the location nor the tracking can be achieved if the nodes that sense the target do not have location information. Therefore, it appears as evident that WSNs need that the nodes they contain know their location information.

Location Discovery (LD), also known as node localization, is the name given to the mechanism or process by which the nodes of a WSN get to know their geographic coordinates, i.e., their location. It is widely acknowledged as being one of the fundamental problems found in the domain of WSNs. We present in this chapter the descriptions and formulations of the LD problem in WSNs most commonly found in the literature. We also discuss basic ranging systems used in WSNs to generate the distance measurements, along with basic position estimation techniques that are used to determine node locations. Other considerations taken into account are error handling and robustness. Finally, the existing literature on research work for the LD problem is reviewed.

9.1 Problem description

LD is widely considered one of the fundamental problems in WSNs ([18, 122, 166]). The objective of LD is to obtain the geographic location of each and every node of a WSN. The first basic intuition is to provide each node in the network with a self-locating hardware, a common example of such hardware would by the Global Positioning System (GPS). However, the use of GPS in all the nodes as an option is generally discarded by a number of reasons:

- **Cost**: the GPS hardware is expensive, while sensor nodes are aimed to be low cost devices.
- **Energy**: the GPS hardware is highly energy-consuming, while sensor nodes are energy-restrained.
- **Size**: the nodes in a WSN have a tight form factor, and the size of the GPS hardware can break it.
- **Requirements**: the GPS location system has a set of requirements for proper functioning, such as outdoor deployment, with line-of-sight to the satellites. The WSN may not guarantee that these requirements will be met in the general scenario.

Instead, the generally adopted procedure is to provide just a small subset of the WSN nodes with self-location capabilities ([141, 208]), in order for them to act as reference points for the rest of the nodes. In fact, LD relies on two basic pillars, without which the problem cannot be solved: the *beacons* and the *references*. The *beacons*, also referred to as *anchor* nodes or *landmarks*, are the subset of the nodes that know their own positions from the start; these locations may have been introduced manually to the nodes (if the nodes were manually placed), or the nodes might be equipped with a GPS-like hardware and hence be capable of deducing their coordinates by themselves. *References* are tuples of the form $(n_i, n_j, \delta_{i,j})$ where n_i and n_j are nodes of the WSN (either one may or not be a beacon), and $\delta_{i,j}$ is a distance measurement such that, for the pair of nodes (n_i, n_j) there exists a distance measurement δ_{i_j}.

The self-location mechanism employed by beacons is out of the scope of our work; hence, we assume that issue to be solved: that is, there is a subset of nodes with self-locating capabilities already present in the WSN upon deployment. We will focus our attention on the rest of the issues related to this problem. This approach for LD is a complex and difficult problem where many additional hardships often arise. The location discovery problem has been proved to be NP-complete ([26, 73, 163]).

We will first give some short definitions. We can assume that locations are given in 2D or 3D. For the sake of simplicity we will assume 2D in the following. Let a and b be two points whose -presumably unknown- coordinates are (x_a, y_a) and (x_b, y_b), respectively, and (x'_a, y'_a) and (x'_b, y'_b) their estimated coordinates (a.k.a. estimated locations, obtained through LD). We define the following:

- Real distance $d_{a,b} = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$,
- Measured or estimated distance $\delta_{a,b}$ (obtained by some measuring technique),
- Calculated distance $c_{a,b} = \sqrt{(x'_a - x'_b)^2 + (y'_a - y'_b)^2}$,
- Measured distance error $\epsilon_{a,b} = \delta_{a,b} - d_{a,b}$,
- Calculated distance error $\epsilon'_{a,b} = c_{a,b} - d_{a,b}$,
- Location errors $\epsilon_a = \sqrt{(x_a - x'_a)^2 + (y_a - y'_a)^2}$, $\epsilon_b = \sqrt{(x_b - x'_b)^2 + (y_b - y'_b)^2}$.

The main objective in the LD problem is to minimize the location errors ϵ_i for all the nodes n_i. However, the values x_i, y_i, are unknown in a real scenario (they are precisely what LD is trying to determine), hence most of the time the proper objective function (or fitness function) cannot be calculated. Therefore, a different function is used to evaluate the candidate solutions; we shall refer to this other function as the *guiding* function; this function is discussed in Section 9.4.

9.2 References generation

As said before, there are two pillars in LD: beacons and references. We refer in this section to the second one, references, and how the nodes in the WSN gain access to them. Since a reference is a tuple $(n_i, n_j, \delta_{i,j})$, nodes n_i and n_j need a way to obtain the estimator $\delta_{i,j}$. There are various estimators that can be used as references, and various techniques by which the nodes can obtain those estimators. The most frequent estimators are:

- **Hop count** ([26, 136, 186]): The simplest available estimator is the minimum hop distance between nodes. This estimator makes use only of the topology of the network, but has low reliability and low accuracy. However, it is always available since nodes have always at least a rough knowledge of their surrounding neighbors, therefore, there is a considerable deal of research done in LD with connectivity information.

- **Distance estimation by hop distance estimation** ([41, 208]): A step further from the hop count technique, this estimator also relies on the minimum hop distance between nodes, but associates a geographic distance to a hop separation. For this, the average hop distance is first determined, using for it the information available: the hop distance and the geographic distance among beacons. Since hops can be highly varying in length, there can be large differences between the calculated average hop distance and the real distance of a hop; furthermore, there can be an error accumulated for multi-hop paths if the differences between the hop distances and the average hop distance are consistent. For instance, the Distance-Vector hop approximation (DV-hop, [170]), uses known beacons to produce average hop distance values, and then uses average hop distance values to estimate the reference distances.

- **Distance estimation by ranging techniques** ([49, 185, 208]): Received Signal Strength Indicator (RSSI or RSS), Time of Arrival (ToA), Time Difference of Arrival (TDoA). These techniques are described in Section 9.2.1. They can use the very radio system used for internode communication, light signal system, or sound/ultrasound system; they are subject to difficulties/noises: interference, shadowing, multipath, environmental variations, etc.

- **Relative position by angular estimation** ([26, 167]): Angle of Arrival (AoA). Measures the angle formed between the node to node link and a reference direction. AoA is normally measured with directional antennas (mobile –rotating– transceiver, antenna arrays), however other methods can be used that do not require additional hardware on the nodes ([167]).

In the problem tackled in the next chapter, the distances are measured with **ranging techniques**.

9.2.1 Ranging techniques

Sensor nodes can measure the distances separating themselves in several ways. This is not a novel feature, since some cellular systems already had this kind of technology. The most widely used techniques are ([204]): Time of Arrival (ToA), Time Difference of Arrival (TDoA), and Received Signal Strength (RSS). We will now briefly describe them.

- In **ToA**, a signal is sent from a transmitter to a receiver. When the signal arrives at the receiver, it in return sends a signal back to the transmitter, who can then measure the time lapse between the first signal was sent, and the second signal was received. Typically an ultrasound is used as the traveling signal and the distance between transmitter and receiver can be estimated as:

$$D = \frac{T \cdot V}{2}, \tag{9.1}$$

  where V is the signal velocity and T is the total estimated signal traveling time. A diagram of ToA is shown in Figure 9.1a. This is the technique that was employed to generate the data we use for this work (Section 10.3).

- The **TDoA** uses two signals traveling at different speeds, such as radio frequency (RF) and ultrasound. The distance can then be calculated from the time lapse between the first signal was received and the second signal was received, as follows:

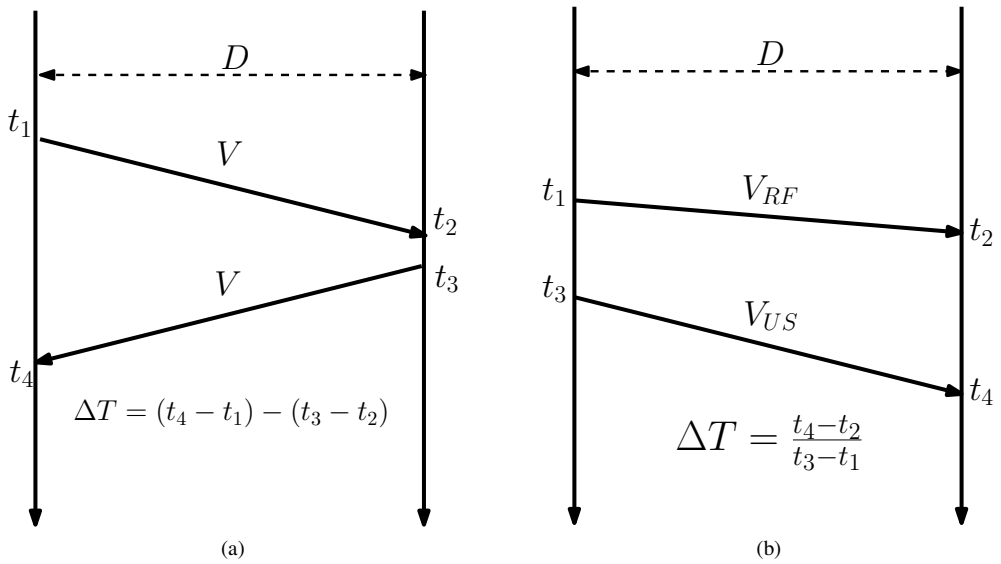


Figure 9.1: Range estimation techniques: (a) ToA and (b) TDoA.

$$D = \Delta T \cdot \frac{V_{RF} \cdot V_{US}}{V_{RF} - V_{US}}, \tag{9.2}$$

where V_{RF} and V_{US} are the traveling speeds of RF and ultrasound signals, respectively. Note that in this case there is no need to divide it in half since signals were only sent from transmitter to receiver, and not back. A diagram of TDoA is shown in Figure 9.1b.

- Finally, **RSS** uses, instead of the signal traveling time, the signal propagation loss as the indicator to estimate the distance separating the two nodes. A signal traveling through space will typically reduce its energy following some law, which can be mathematically modeled. A widely spread model for the path loss is the following:

$$PL(d) = PL(d_0) + 10n \log\left(\frac{d}{d_0}\right), \tag{9.3}$$

where $PL()$ is the path loss exponent function measured in decibels, d_0 is a reference distance, and n is an exponent that depends on the environment (generally ranging from 2 to 4). An illustration of the RSS over distance is shown in Figure 9.2.

It is widely assumed that RSS is the method that incurs the most significant errors, since path loss is subject to quick and large variations such as shadows or fading ([182, 194]). However, RSS is an interesting method since it can be easily implemented in sensor nodes without requiring any additional hardware, using the communications subsystem, which results in a virtually zero-cost method.

- Additionally, the **Angle of Arrival** (AoA) technique allows two nodes to determine their relative positions in terms of direction (but not distance). Although AoA technically speaking does not constitute a *ranging* technique, we include it in this section for the sake of completeness since it offers an alternative way of producing references (of its own kind). The AoA approach requires an array of receivers, which can determine the direction of the incoming signal. Some approaches propose

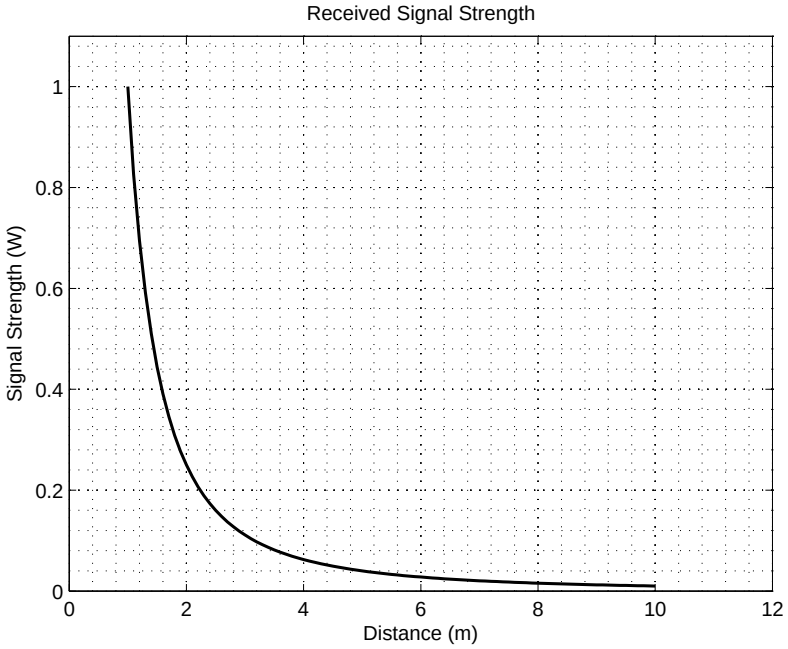


Figure 9.2: Range estimation techniques: Received Signal Strength Indicator (RSS or RSSI).

that only beacons use dynamic directive transmitters that transmit a beam describing a periodic rotation ([167]), and thus nodes can determine their direction from beacons just by checking the signals arrival times (assuming that the signal traveling latency is negligible compared to the rotation period of the beam).

9.3 Position estimation techniques

Since LD is one of the main problems found not only in WSNs, but also in many other wireless networks (cellular networks or MANETs, for instance), there are several techniques that have been defined to solve the canonical LD problem.

- **Trilateration** ([139, 182]). This is the basic approach for single point localization, typically used in combination with ToA-like ranging systems; three distance references with beacons are required for this method, and the location is determined as the intersection point of the corresponding circles. A simple case of trilateration is shown in Figure 9.3a, where the position of point P is obtained using reference distances R_1, R_2 and R_3 to three beacons.

- **Multilateration** ([141, 182, 208]). Multilateration is often used with a TDoA-based distance measurements; in multilateration, three or more landmarks receive/send the signal from/to the object to be located in a synchronized fashion. Based on the registered time differences, the differences in distance are known. From that information, the position of the object can be estimated.

- **Triangulation** ([132, 167, 182]). Triangulation solves the location problem by using triangular geometry (trigonometry) with angular references with respect to beacons: the crossing point of the defined lines is the location. A simple case of triangulation is shown in Figure 9.3b, where the position of point P is obtained using angular references α and β with two beacons.

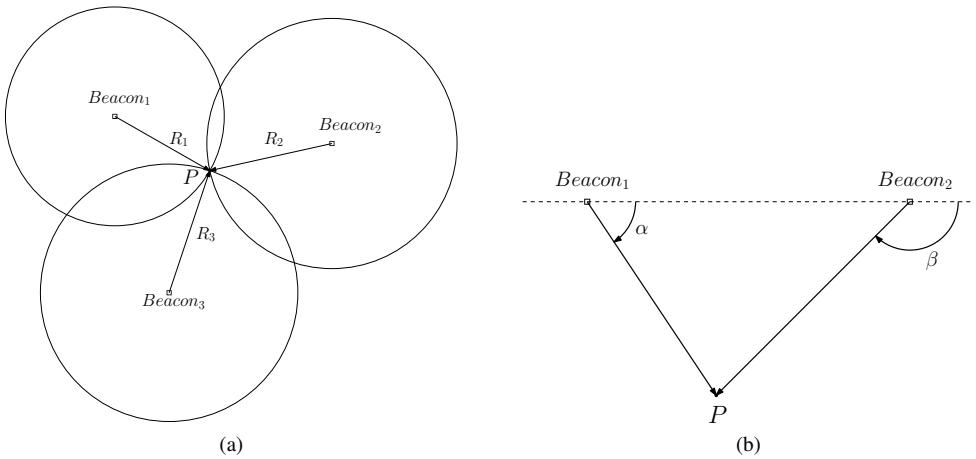


Figure 9.3: Atomic localization techniques: (a) trilateration and (b) triangulation.

- **Multidimensional scaling** (MDS, [49, 182, 186]). An MDS algorithm starts with a matrix of item-item similarities, then assigns a location to each item in N-dimensional space, where N is specified a priori. MDS is often used as a range-less location system, when only topological information (which nodes are connected, which nodes are not) is available, and no ranging information is available.

The location techniques described above are referred to as *atomic*, since they can only locate a single target at a time, and for this they require at least three references with as many beacons. Therefore, using strictly atomic localization, it is impossible to locate nodes that have less than three references with beacons. Thus, several more advanced techniques have been proposed to overcome this limitation:

- **Iterative localization** ([182]). In this approach, nodes are iteratively converted into beacons as they are located; thus, the number of beacons progressively increases, and at the same time more nodes can be located as they gain access to three (or more) beacon references. For instance, in Figure 9.4a, only node 1 has three valid references and can be located. But if upon localization node 1 becomes a beacon, then node 2 gets three valid references. If the process is repeated, nodes 3, 4, and 5 can be localized in turn.

- **Cooperative localization** ([182]). This technique goes a step further than the previous one; in cooperative localization, a connected set of nodes can be located as long as there are three beacon references to nodes of the set. The procedure consists in defining the equation system corresponding to all the references in the subnetwork (both beacon-node and node-node), and simultaneously obtaining (or iteratively refining) the nodes positions. Figure 9.4b shows an example case where neither node 1 nor node 2 can be located, and an iterative localization cannot solve the situation; however, if the full system is defined with two unknown variables corresponding to the locations of nodes 1 and 2, it can be solved and both locations are obtained. If the resulting equation system is well defined, it can be entirely solved with matrix algebra.

9.3.1 Coping with errors in the measurements

Unfortunately, most of the ranging systems used in WSNs have non-negligible measurement errors; that is, even if generally $\delta_{a,b}$ approaches $d_{a,b}$, still $\delta_{a,b} \neq d_{a,b}$. Due to this, the references are not exact values, only approximations. Therefore, the trilateration, multilateration, or triangulation systems cannot be solved

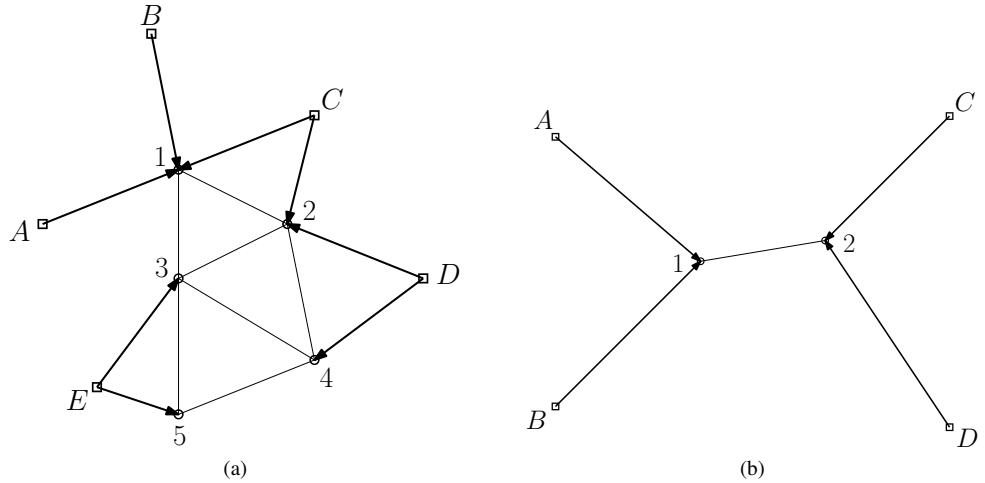


Figure 9.4: Advanced localization techniques: (a) iterative localization and (b) collaborative localization. Beacons are named with letters, and regular nodes are numbered.

exactly, and matrix algebra can no longer be an option. When measurement errors are an important factor, the following procedures can be employed:

- **Mass-spring relaxation** ([26, 136, 138]). The mass-spring relaxation allows deviations in the inter-node distances with respect to the measured values, following a physical spring model. For each reference link $\{i, j\}$, a virtual force is associated as $F_{i,j} = d_{i,j} - c_{i,j}/d_{i,j}$. Thus, the link distances are modified until the resulting system stabilizes, that is, the sum of forces applied to each node by its associated references equals zero. A general approach to this is to define the system energy $E = \sum F_{i,j}^2$, and iteratively modify the node positions according to the forces acting upon them, until the energy ceases to decrease.

- **Minimize the Mean Square Error** (MMSE, [142]). Similar to the previous one, it also considers the deviation of the reference links lengths. Assuming that the measurements are fairly accurate, the aim is at minimizing some estimation of the error defined from that distance deviation. The most common approach is to use the square norm $\sum(\delta_{i,j} - c_{i,j})^2$ as the function to minimize. Many optimization methods can be used in this case, from local search methods to metaheuristics. Additionally, this method can be enhanced to integrate problem knowledge (and increase its robustness and accuracy), by weighting the associated error of each link by a confidence factor on the link measured distance accuracy: $\sum \omega(\delta_{i,j}) \cdot (\delta_{i,j} - c_{i,j})^2$.

- Obtain the **Maximum Likelihood** locations ([73]). This option can only be used if a probability density function (PDF) of the measurements is available (hence requires problem knowledge). This method is computationally costly, but copes with measurement errors in a natural way, and has been proved to obtain accurate results. This method is further explained in Section 9.4.

Other conceptions consider that errors in ranging, either for time-based or received signal strength, lead the nodes to estimate larger distances. This is due to the fact that both signal attenuation, or indirect path effect, induce the nodes to have errors by excess. Therefore, an approximation that mitigates the negative effects of distance estimation errors, assuming that these errors will be excess errors, consists in treating the estimated distances as upper bounds of the real distances, instead of as accurate estimations. Examples of this are in [208]. Another system is the voting system; in it, each landmark *votes* for a crown region

around itself with central radius equal to the measured distance and width relative to the expected error; the centroid of the most voted region is selected as the estimated location ([142]).

9.4 Guiding functions in LD

Differently from the other problems solved in this thesis, the LD problem has a special characteristic that renders it specially difficult to solve: the optimization objective of the problem cannot be evaluated during the optimization process. That is, the *guiding* function and the *fitness* function are different; in fact, in a real scenario, the fitness function is unknown and can never be evaluated (since it requires knowledge of the optimal solution).

Therefore, a guiding function has to be defined for LD such that it is computable from the available information: the beacon locations, and the inter-node distances (real node locations are unknown and cannot be used). The most widely used guiding functions are the following:

- **Error norm functions** ([73]). Belonging in this first category are the methods that assume that the measured distances are the real distances, and solve the optimization problem by minimizing the calculated distances error norm $L(c_{ij} - \delta_{ij})$. Typical norms are L_1, L_2, or L_∞ ([123], equations 9.4, 9.5, and 9.6, respectively). This method can produce good results when the measurements are accurate (hence the assumption can be considered as mostly correct), but will lead to large positioning errors if the measurement errors are large. Although these methods do not require any specific problem knowledge besides the instance data, there are several ways in which it could be introduced in order to tune the method. Among them, we could mention a weighted norm minimization, in which measured values are weighted according to their expected reliability, which in turn is obtained from some problem knowledge.

$$L_1 = |\epsilon_1| + |\epsilon_2| + |\epsilon_3| + ... + |\epsilon_L|, \tag{9.4}$$

$$L_2 = \sqrt{\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + ... + \epsilon_L^2}, \tag{9.5}$$

$$L_\infty = max\{\epsilon_1, \epsilon_2, \epsilon_3, ...\epsilon_L\}. \tag{9.6}$$

- **Likelihood functions** ([73]). Methods in the second category require previous knowledge on the measurement errors, since they use a PDF $P(\delta, d)$ for the measured distance (δ) vs. the real distance (d). By replacing in the formula real distances by calculated distances (c), the global likelihood value for the WSN node positions, calculated as the product combination of the probabilities for every c_{ij} and δ_{ij} (Equation 9.7), should be maximal for the values that match the definition, i.e., when the calculated distances equal the real distances: $c_{i,j} = d_{i,j}$.

$$ML(s) = \Pi_{i,j \in s} P(\delta_{i,j}, c_{i,j}). \tag{9.7}$$

9.5 Additional considerations

The LD is a particularly difficult task in WSN for a number of reasons, among which we can list the following:

- The ranging process incurs high energy expenses, thus has to be restricted to the bare minimum.
- Radio-based ranging can use the integrated radio system of the sensor node, but has very low accuracy ([233]). More accurate ranging systems require additional hardware and are often unavailable.

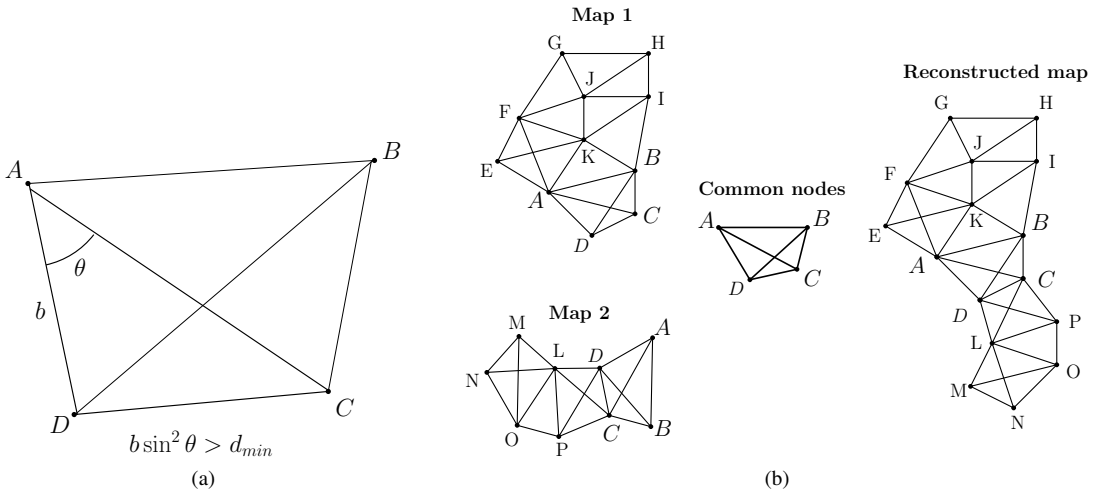


Figure 9.5: Other localization techniques: (a) robust quadrilaterals, (b) MAP localization.

- The errors in ranging can be very large (even larger than the real value), and are very hard to model with some standard (e.g., Gaussian) model. Generally, non-parametric ad hoc models need to be set up, based on measurement data ([73]).
- The real fitness function is unknown, thus the guiding function and the fitness function are different functions.

Robustness in LD is a desirable property, and many techniques have been devised to ensure or optimize the robustness of the solution. By robustness, the common understanding is that solutions should not incur any large error with respect to the true node locations; this is typically reduced to obtaining a solution with the correct general layout of the network, that is, a solution that has no flip or rotation errors. In the following, we shall refer to all these kind of errors (including both flips and rotations) as *flip errors*. A method to enhance the robustness is to study the rigidity of the network ([136]), that is, the capacity of the nodes to move in space without violating the distance constraints. Ideally, the network should be rigid, hence the nodes should not be able to move while respecting the distance measurements.

Some approaches of LD focus specially on the robustness of the localization process, taking into account the facts that the available references may be scarce, or that the estimated distances may contain (sometimes large) error components. For instance, in [163] the robustness of the location process for each node is evaluated with *robust quadrilaterals*: a quadrilateral that can be realized without ambiguity; a node is only located after it becomes the vertex of a robust quadrilateral whose other vertices are located, thus flip ambiguities are avoided. A robust quadrilateral has 6 distance references, the 4 edges and the 2 diagonals, and can be decomposed in robust triangles, where there are no short sides or small angles: $b\sin^2\theta > d_{min}$, where b and θ are the smallest edge and angle, respectively, and d_{min} is a threshold value defined depending on the expected measurement errors (see Figure 9.5a). In an alternate definition, a robust triangle is one in which all the angles are larger than 30^o ([139]). The algorithm proceeds by constructing an initial robust quadrilateral, then expanding it by adding nodes that form new robust quadrilaterals with previously included nodes. The nodes that cannot be located without risk of flip are left unlocated, thus sacrificing location information for the sake of robustness.

Other approaches aim to divide the problem in order to solve it in a distributed fashion; these approaches are usually nicknamed *MAP* approaches ([138, 186]). They divide the WSN into smaller sets of nodes (quite like neighborhoods) called *maps*, then solve the LD problem –usually producing virtual coordinates– for

each map. Then the different maps are iteratively fused by using at least three shared nodes until ultimately a single map is obtained. Figure 9.5b illustrates the fusion process of two maps. Finally, absolute coordinates can be obtained if at least three beacon nodes are present in the final map. A different distributed strategy is that of iteratively growing a map of virtual coordinates starting from an initial robust triangle ([139]), and iteratively adding nodes such that each newly added node forms a robust triangle with two nodes previously located (see highlighted triangle in Figure 9.5a).

Finally, another important concern in LD is **indicator consistency** between the guiding function and the fitness function. Or put in a different way, the answer to the question: does the guiding function have the same optimum as the fitness function? By consistency, we refer to the fact that if a given solution s_a has a better fitness value than s_b, then the guiding function should also prefer s_a over s_b. Since fitness and guiding functions are different functions –out of necessity–, the previous correspondence does not happen always; hence, the ratio of solution pairs that are correctly discriminated by the guiding function with respect to the fitness function is the *consistency* of the guiding function.

9.6 Literature review

Location Discovery is one of the most prolific topics in the domain of WSN. As such, there is a extensive literature concerning LD, of which we will make a short review. Our purpose is to display the main aspects involved, and the main approaches adopted by the research community to tackle this fundamental task.

An interesting early survey of the techniques used in WSNs for LD can be found in [182]. In this work, the main ranging techniques (RSS, ToA, TDoA, AoA) as well as the main location techniques (trilateration, multilateration, triangulation, maximum likelihood) are described. Additionally, the authors present the Medusa nodes (see Section 2.3) for ultrasound ranging, and their performance is assessed. In [81], the authors present a short description of an acoustic range estimation device. The range estimation is based on the ToA of an acoustic signal, called chirp, between two nodes; the performance is enhanced by the use of broadband techniques.

Research on LD has gone into many different directions. In fact, not all research is about *solving* the problem; for instance, some works are focused on analyzing the properties and inherent difficulties of LD, mostly error-related (error factors, nature of errors in location, effect of location errors, etc.). A theoretical study on the Crámer-Rao Lower Bound on positioning error was performed in [171]. In that work, the authors consider the range-free location system based on hops from the landmarks (anchor nodes), and the Distance Vector-position method, in which both distance and angle estimates are available by nodes. Finally, some conditions are given under which DV-position outperforms range-free location. A study on the location errors in several applications for WSN is done in [194]. The work focuses on exposure, best- and worst-case coverage, and shortest path routing. The norm functions are used in this work as the objective functions for location discovery. The process is incremental: at each step, all nodes that can triangulate their locations using distance measurements from beacons determine their locations, and become beacons themselves for the rest of the nodes. The sources of errors are identified and modeled, and the propagation and effect of the errors are studied. The effect of radio irregularity on RSS-based location systems is studied in [233]. The authors define three parameters to characterize their proposed Radio Irregularity Model (RIM): the Degree of Variance (DOI), the Variance of the DOI (VDOI), and the Variance of Sending Power (VSP). The DOI represents the maximum received power variation per angular displacement; the VDOI represents the maximum relative variation in DOI between two nodes; the VSP represents the maximum relative transmitting power variation between two nodes (due to hardware differences). Some security issues related to LD, such as possible attacks, have also been widely considered. Possible security threats related to LD, such as Sybil or wormhole attacks, are described in [132] and [143]. Some techniques for attack resistant location discovery are described in [142], which are based on defining data-driven bounds on the accepted mean square errors (as in the L_1 norm) for every single node positioning, and based on voting. A similar method combined with the use of difectional antennae is the method proposed in [132]. The focus

adopted in [143] is different, where the authors suggest a method for detecting malicious beacon nodes based on a combination of wormhole detectors with "detective" beacon nodes. In [41], the performance of localization algorithms using signal strength as the ranging technique in the case of signal strength attacks is evaluated. The authors distinguish two kinds of location techniques: point-based, that return a single point location for each node, and area based, that return a location area for each node. The experiments are performed indoor, thus multilateration cannot be performed, and a radio-map is established instead. Algorithms of the first class are Radar and Highest Probability, and of the second class are Simple Point Matching (SPM), Area Based Probability, and Bayesian Networks. Both attenuation and amplification attacks are considered, each attack may be done over a single landmark, a set of landmarks, or all landmarks. In [185], the objective is to produce LD problem instances that are difficult to solve. Several parameters are identified as having an influence on the difficulty of the LD; among them are the number of nodes, number of beacons, average graph connectivity degree, average signal noise, etc. Using a set $\mathcal{M}$ of experimental distance measurements $m_{i,j,k}$ where i and j are node indexes and $m_{i,j,k}$ is the k^{th} measurement for that node pair, producing a problem instance amounts to selecting a subset $\mathcal{S} \subseteq \mathcal{M}$ such that some node pairs exist in the subset, and for each of them only one measurement is selected. The difficulty of an objective function for the produced instance is measured from three indicators: rank-based consistency, variance, and drifting.

Among the works oriented towards solving LD, there are a number of trends. The main classification can be established depending on the information used as input. The first category contains techniques that do not make use of distance estimations, and generally rely on connectivity information uniquely. An MDS algorithm to solve the LD problem is used in [186] using only connectivity information. The authors propose two approaches, one centralized (MDS-MAP(C)) that builds a global map, and a distributed approach (MDS-MAP(P)) that divides the WSN in small maps, solves the location for each map, then rebuilds the global map by fusing smaller maps. The distributed version has the advantage of being less sensitive with respect to shortest path-based distance estimation accuracy (since shortest path is generally not a reliable estimation of inter-node distances in non-homogeneous WSNs, algorithms should not rely on it). Both algorithms have a basic version, and a refined version that includes a least-squares refinement phase. The authors of [136] present a method to solve LD relying solely on connectivity information, with special focus on complex terrain shapes, which are difficult scenarios for LD in which flips often occur. The authors rely on the rigidity of the graph to avoid flip errors. The key idea is to identify a subset of the nodes that are placed along the region frontiers (external and internal); there are several distributed techniques that do this. This subset will act afterwards as the reference set, or landmarks, for the rest of the nodes to locate. The whole LD process works with virtual coordinates, since there is no predefined set of GPS-equipped beacons. A beacon-less scheme using a Maximum likelihood approach to solve LD is presented in [72] that uses only connectivity information. The authors argue that indeed absolutely removing the need for beacon nodes can be interesting from an economic perspective. In their problem formulation, the nodes must have previous knowledge about their intended deployment area (a group of nodes may be dropped from a plane over location 'X'), and discover their location by analyzing the drop locations of their neighbors using ML. A gradient descent and a geometric approach are proposed to solve the problem. Some works use angular information either in addition to connectivity or alone. The LD problem based solely in connectivity and angle information, and without beacon nodes in the network, is solved in [26], with the purpose of using it for GPSR routing; additionally, a planar spanning of the graph is obtained by using exclusively angular information. The Unit Disk Graph embedding is solved using Linear Programming, with links and loops as constraints. To reduce the number of variables, geometric techniques are proposed: apply the proportion ratios of closed triangles to reduce three link variables to a single one whenever possible. Another angular approach to the LD problem (AoA) is presented in [167]. In this work, nodes do not require specific hardware, since they receive the signals from the beacons and only register the time separations among them. Beacons, however, require specific hardware to transmit a rotating beam-shaped signal. The authors claim their system achieves higher precision that RSS-based systems, and identify two sources of error: beam width and multipath. They tackle the first by choosing the time instant when the received signal

has maximum strength, and the second by performing multiple locations with as many different sets of 3 beacons.

The second category of LD problems uses distance estimations as references. Many different solving methods have been proposed for these problems, which can be solved with greater accuracy than the previous ones. Some early works on the subject employ the norm functions L_1, L_2, and L_∞ as the fitness functions for the optimization procedure. In [123], all three functions are alternatively used in addition to a location error minimization function $\sum_{j=1}^{n_B} \sqrt{(x_{AIj} - x_{AFj})^2 + (y_{AIj} - y_{AFj})^2}$, where AIj represents the original GPS-determined position of the beacon j, and AFj represents the location determined for that beacon. The paper contemplates the existence of Gaussian error both in the GPS locations of the beacon nodes, and in the distance measurements (with standard deviation proportional to the real distance). Surprisingly enough, L_∞ outperforms the other two norm functions in the experiments performed. Maximum likelihood is used to solve the LD problem in [73], with a strong focus on the statistical modeling of the measurements. Real data by sensors with acoustic ranging based on TDoA are used. Several families of error models are tested, and compared against kernel-smoothing. Off-line and on-line constructed models are used. In [49], the LD problem is formulated as a Multidimensional Scaling problem (MDS). Formally speaking the MDS problem uses a number of dissimilarities (i.e., distances) in order to determine the multidimensional values (i.e., coordinates) of a set of objects. The authors propose a distributed iterative algorithm in which every node refines its location by using location information and measured distances from its neighbors. Special stress is put on the neighbor selection as a two-stage resolution process is used to eliminate the induced bias. The Curvilinear Component Analysis (CCA) technique is presented for LD in [138]. The distributed version, CCA-MAP, is analogous to MDS-MAP: the LD problem is solved for small subnetworks producing small maps of virtual coordinates, which are progressively fused into larger maps, until a single map is ultimately obtained. Each fusion of two maps requires three shared nodes between the maps. Finally, with three beacons, the global map is transformed into absolute coordinates. The key idea of CCA is to use data projection to reduce the dimensionality of a vector system, such that the system of reduced vectors maintains the inter-vector distances. This is applied by initially generating expanded point coordinates (with more dimensions than the real points), that respect the inter-point distances corresponding to the measured distances, and, through an iterative process, obtain their projection transformation: the real point coordinates. A different focus is adopted in [163], where the main stress is put on the robustness of the localization. A robust localization is defined as one that avoids flip ambiguities. A distributed algorithm for beaconless network localization is proposed, in which nodes use noisy distance measurements only. Thus, locations are determined up to a global rotation and translation. The concept of *robust quadrilaterals* is introduced, representing quads of nodes that can be unambiguously located even in the presence of measurement noise; when a node cannot be included in such a quad, its location is not determined and is considered unknown. The algorithm is implemented on a physical WSN, and supports localization of mobile nodes. The Ad hoc Positioning System (APS) is proposed in [170], and its performance is tested. The algorithm uses estimated distances from the nodes to the different landmarks. Since not all landmarks are within ranging distance from the nodes, the authors explore the use of three approximate distance propagation techniques: the Distance Vector hop (DV-hop), where the number of hops is multiplied by the average hop length; the Distance Vector distance (DV-distance), where the cumulative measured hop lengths are used; and the Euclidean propagation method, where node A requires distance references from two nodes B and C from the landmark, along with estimations of the distances AB, AC, and BC, to estimate its own distance.

Finally, some works are specifically designed to cope with the main difficulties found in LD. For this, the generally adopted method is to tackle problem instances that are known for being difficult to solve. The problem of LD inside concave spaces is considered in [208]. In this problem formulation, the nodes of the WSN use their available references with respect to the beacons when they can directly estimate the distance, or by using a hop-distance approximation. In a concave area, the estimated distances may be much larger than the real ones and thus produce errors; the proposed solution consists in considering the estimations as *upper bounds* of the distances, instead of approximations of the distances. This way, each reference is not

treated as a circle (trying to approach the node to the circumference), but instead as a full disk (anywhere inside the disk is a valid location). The approximated location is the intersection of all reference disks. Some enhancements are proposed, as an iterative multilateration version where nodes may only locate themselves if the expected accuracy surpasses a threshold, and then become beacons that can be used by unlocated nodes. LD in anisotropic networks is studied in [141]. The concept of an anisotropic network is a network in which the connectivity properties of the nodes are not homogeneous over all the space occupied by the WSN; for instance, a network whose nodes are not uniformly distributed is anisotropic, like a network with an inner hole. Another example of anisotropic network is a network in which R_{COMM} depends on the node location. The authors propose the use of Proximity Distance Map (PDM) for LD in these networks. In this technique, nodes use a proximity measure towards the beacons; using the proximity measures among beacons, the transformation matrix T is generated such that the proximity measures of the nodes can be transformed into geographic distances, and the multilateration system is obtained. The work presented in [139] focuses on LD for networks with irregular shapes, for instance a *C* shape or an *O* shape, for which most techniques initially proposed for LD do not perform well. Their proposal relies on network partition into several localized subnetworks, locating nodes in the subnetworks, then reconstructing the global network. In this technique, nodes search their neighborhoods to form robust triangles, that become beacons for the rest of the nodes; then the rest of the nodes try to localize themselves using three non-collinear beacon references. The global map of virtual coordinates finally gets absolute coordinates using at least three landmark references.

Our interest lies with the resolution of a generic approach to LD, instead of specific *hard* instances with weird network shapes (which may be unlikely to happen in a real case). In the problem we consider the hardship comes from the use of **real**, error containing measured distances from which we define the instances. Since we propose the use of metaheuristics to tackle this problem, we are left with two main types of guiding function, namely the error norm functions and the likelihood functions. In the next chapter, we will perform a study on the two types of function, and propose a solving method using a combination of them.

9.7 Conclusions

In this chapter, we have presented and described the Location Discovery problem in WSNs. LD is widely acknowledged as one of the most prominent problems found in most ad-hoc networks, and notably in WSNs; it has also been proved to be NP-hard. In short, the LD problem amounts to finding the geographic coordinates of the nodes of a WSN with the minimum error possible, given a set of internode distances and beacon node references.

We have briefly presented the main methods used to measure internode distances: received signal strength, time of arrival, and time difference of arrival; we have later described the basic location methods that have been used in the literature: trilateration, multilateration, triangulation, and multidimensional scaling. Also, some simple enhancements as iterative or cooperative localization have been explained. Then, we have highlighted the importance and effect of errors in measurements, and presented the two main types of guiding functions that may be used to solve LD as an optimization problem: error norm functions and likelihood functions.

Finally, we have provided a review of the existing literature in the field, with special attention to the main difficulties found in LD. In the next chapter, we will perform a study on the two types of guiding function and propose a new solution method combining both, that will be tested experimentally on real data instances.

Chapter 10

Resolution Methodology and Results for Location Discovery

In the previous chapter, we presented the Location Discovery (LD) problem in WSNs, the techniques used to generate the required data (distance estimations), some solving procedures, and the main difficulties found in this problem (errors in measurements, finding a guiding function, etc.). In this chapter we describe the formulation adopted for the LD problem and present the real data used to define the problem instances that are solved in this thesis. Additionally, we study the two main types of guiding function used to solve the LD problem, error norm and likelihood functions, and propose a two-phase resolution procedure that combines both.

We adopt a mono-objective approach to solve this problem, and try three different types of algorithm to solve it: a trajectory-based metaheuristic (SA), an evolutionary algorithm (GA), and a particle swarm based algorithm (PSO). The representation used for the solutions and the operators used by the different solving techniques to manipulate these solutions are described. The instances solved are generated using real data measurements organized into 33 datasets taken over the course of a few days; the size of the corresponding WSN is of a hundred nodes approximately.

Using the available problem data we conduct a study of the consistency of the two main types of guiding function used in LD: error norm (we pick L_1), and likelihood (we pick a pyramid kernel function). This study serves as the basis for the proposal of our novel approach: the combination of the two types of function into a sequential two-stage approach. Then, we select 10 data sets to be the test instances, where we test the effectiveness of the enhancements proposed for the error norm function, the expected accuracy of the locations depending on the number of beacons, and the relative performances of the different algorithms. Finally, we assess the validity of our two-stage proposal by comparing the results it obtains with the results obtained by each of the two types of guiding functions working separately.

10.1 Problem formulation and models

The LD problem was previously presented in Section 9.1. We briefly review and complete the definition in this section. Our formulation of the problem is as follows: given a set S of N nodes $s_i, 1 \leq i \leq N$, where the subset $s_j, 1 \leq j \leq K < N$, has previously knowledge about their locations (anchor nodes), and given a set of distance estimations $\delta_{i,j}, 1 \leq i, j \leq N, i \neq j$, we have to determine the locations of every node $s_l, K < l \leq N$, such that the location error is minimized. Hence, we define the *fitness* value of a candidate solution as the average position error of all the nodes (see Equation 10.1). More formally, let $\vec{x}$ be a given solution to the LD problem, in which x_i represents the estimated location of the i^{th} node in the network, and let n_i be the *real* location of the i^{th} node, then we have:

$$fitness(\vec{x}) = \frac{\sum_{i=1}^{|\vec{x}|} |x_i - n_i|}{|\vec{x}|}, \tag{10.1}$$

where $|\vec{x}|$ is the number of elements in $\vec{x}$, that is, the number of nodes in the WSN.

As was argued in Section 9.1, the real node locations are unknown in a practical scenario (otherwise there would be no need to solve the LD problem), thus the fitness function cannot be computed. Instead, a guiding function is defined as the function to be optimized; this function needs to be such that solutions close to the optimum of the guiding function should also be close to the optimum of the fitness function as well. We consider two possibilities for the guiding function of the LD problem:

- An **error norm** function of the produced distances over the measured distances. When an error norm function is used, the underlying assumption is that the measured distances are fairly accurate, and that producing a WSN in which the inter-node distances match the measured distances results in trustful node locations. They are the simplest guiding functions for LD, and do not require previous knowledge about the error model. The norm functions are a family of $\mathbb{R}^n \rightarrow \mathbb{R}$ functions that serve as indicators of *how much error* a given candidate solution incurs, in terms of inter-node distances. When an error norm function is employed the location discovery becomes a *minimization* problem.

  The most popular norm functions are L_1, L_2, and L_∞ ([123, 194]), which were shown in equations (9.4), (9.5), and (9.6), respectively. In the absence of measurement errors, the LD problem can be solved optimally using any of these norm functions as guiding function. However, in the presence of significant measurement errors, the performance of any search algorithm that uses an error norm function is severely degraded. When the errors are highly varying (as is usual in WSN) the L_1 norm produces the best results, while L_∞ produces the worst ones. In [73], all three norm functions were tested on a small instance where 1 node was located using 9 anchor nodes as references, with node to node distances ranging from 7 m to 45 m. The location errors obtained were 1.272 m, 5.737 m and 8.365 m for L_1, L_2, and L_∞, respectively. Therefore, we set L_1 as the base error norm function in our work.

  Let $l_i, 1 \leq i \leq L$ be the measured link distances ($l_i = \delta_{i1,i2}$ corresponds to the link between nodes $i1$ and $i2$). We define the measured distance error for link l_i as $\epsilon_i = c_{i1,i2} - \delta_{i1,i2}$. The simplest approach to the use of a measurement error model is to incorporate a weighting function to the norm operator. This means we will still use the norm function, but we will multiply every link distance error ϵ_i by a weight ω_i that indicates how reliable the measured value is. The resulting modified error norm function, L_1', is shown in Equation 10.2.

$$L_1' = |\omega(\delta_1) \cdot \epsilon_1| + |\omega(\delta_2) \cdot \epsilon_2| + |\omega(\delta_3) \cdot \epsilon_3| + ... + |\omega(\delta_L) \cdot \epsilon_L|. \tag{10.2}$$

  The weighting function is defined based on knowledge of the measurement errors. It qualifies the reliability of a measured distance δ, in such a way that a distance measurement that is believed to be correct will receive a corresponding large weight, while one that is suspected to be incorrect will receive a corresponding low weight. This way, the weighting function should help reduce the impact of measurement errors. For our work, the weighting function employed is calculated as follows:

$$\omega(\delta) = \exp\left(-\frac{average\ location\ error(\delta)}{NORM}\right), \tag{10.3}$$

  where δ is the measured link distance in meters, the average location error is calculated for all links in the data base ranging from $(\delta - margin)$ to $(\delta + margin)$, and $NORM$ is a normalization value. Both $margin$ and $NORM$ are determined empirically.

- A **likelihood** function of the real distances over the measured distances. The likelihood function relies on the use of a probability density function (PDF) that characterizes the measurement errors, thus natively incorporates the use of measurement error knowledge. In this case, a full measurement error model has to be developed such that for any pair of real and estimated (measured) distances (d, δ), the model provides the likelihood (probability) that given an estimated distance δ, the corresponding real distance is d, noted $P(d, \delta)$. We say that every link has an associated probability, or likelihood. In this case the LD becomes a *maximization* problem, namely the Maximum Likelihood problem (ML), where the value to be maximized is the global likelihood L (Eq. 10.4).

$$L = P(d_1, \delta_1) \cdot P(d_2, \delta_2) \cdot P(d_3, \delta_3) \cdot ... \cdot P(d_L, \delta_L). \tag{10.4}$$

There are many possible manners to create a statistical model for the measurement errors. One can assume the measurement errors follow some probability distribution (Gaussian, beta, gamma, etc.) and adjust the corresponding parameters to best fit the available data. These models are called parametric models. Rather than the previous, we use a non-parametric kernel error model similar to the one in [73]. In this model, each measurement is represented by a surface function called kernel function, centered at the (x, y) position corresponding to the pair of *real* and *measured* distances for that measurement. We have chosen a pyramid function as the kernel function (see Figure 10.1). The complete PDF function is obtained by *adding* the kernels corresponding to all the measurements available, and normalizing the resulting function; this way, a set of discrete points in 2D is transformed into a continuous 2D PDF function.

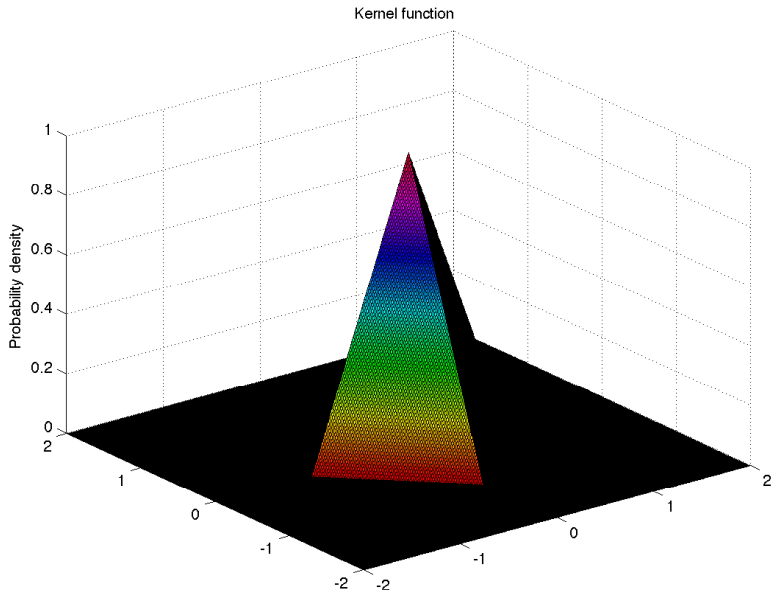


Figure 10.1: Kernel function for the probability density function in LD.

10.2 Representation and operators

In this section we describe the representation used for the candidate solutions for the LD problem, and the way they are manipulated by the different genetic operators used with these solutions: mutation, crossover, and flight operator of PSO.

10.2.1 Solution encoding

In LD, a candidate solution is an array of 2D coordinates indicating the locations of the nodes in the network. First, every node in a WSN is numbered. Then, we use a straightforward encoding: an array of real numbers with length double the number of sensor nodes in the WSN. The first two elements are respectively the x and y coordinates of the first node, from this point on every two values represent the x and y coordinates of one of the following nodes, respectively. Figure 10.2 displays the solution encoding for this problem. The positions corresponding to the beacon nodes can take arbitrary values, since they are not computed[1].

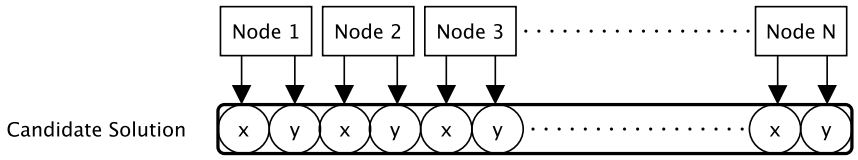


Figure 10.2: Solution encoding for LD.

10.2.2 Genetic operators

SA, GA, and PSO are the algorithms that were used to solve the LD problem, hence the genetic operators involved are mutation, crossover, and PSO's specific operators.

Mutation operator

The mutation operator is employed in both SA and GA. We use an adaptive mutation operator. This mutation selects each of the individual coordinates independently with a given probability p_m. Figure 10.3 illustrates the mutation operator used for LD.

When a coordinate has been selected, it is modified by adding a displacement d, which is a random value between $-R_{max}$ and $+R_{max}$. The value for R_{max} is selected as the average error per link measurement (considering L_1 norm) multiplied by a scaling factor we refer to as *mutation intensity*, regardless of the chosen guiding function. The intuition behind this is that when the error value is low the solution is close to the optimum, thus smaller steps should be used, while when the error value is high the solution is far from the optimum, and larger steps are preferred. Additionally, this value is further weighted by a value representing the algorithm's execution progress: $1 - evaluations/max\ evaluations$. The idea behind this is to have an increasingly fine grain precision, by performing smaller steps, even when the error norm is lower bounded.

Crossover operator

The crossover operator is used in GA. For LD, the crossover operator used is the Simulated Binary Crossover (SBX) crossover, a well-known operator ([58]).

[1] These values are still included for code simplicity

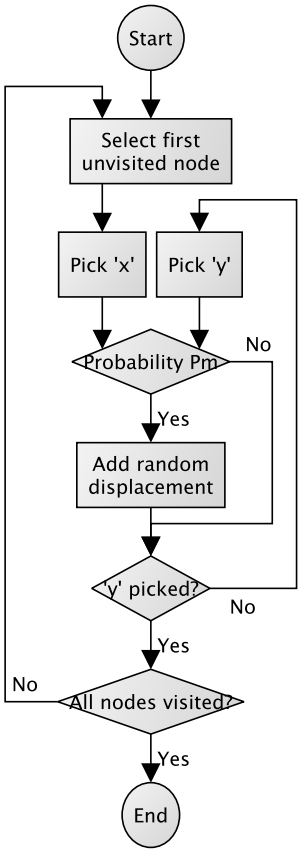


Figure 10.3: Mutation operator for the LD problem.

Flight operator

The flight operator is used by PSO to explore the search space, by updating the particles. The way in which PSO updates the particle $\vec{x}_i$ at the generation t is given by the formula:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t), \tag{10.5}$$

where the factor $\vec{v}_i(t)$ is known as velocity and is calculated as:

$$\vec{v}_i(t) = w * \vec{v}_i(t-1) + C1 * r1 * (\vec{x}_{pbest_i} - \vec{x}_i) + C2 * r2 * (\vec{x}_{gbest_i} - \vec{x}_i). \tag{10.6}$$

In this formula, $\vec{x}_{pbest_i}$ is the best value that $\vec{x}_i$ has ever had (personal best), $\vec{x}_{gbest_i}$ is the best particle (also known as the *leader*) that the entire swarm has ever viewed (global best), w is the inertia weight of the particle and controls the trade-off between global and local experience, $r1$ and $r2$ are two uniformly distributed random numbers in the range $[0, 1]$, and $C1$ and $C2$ are specific parameters which control the effect of the personal and global best particles.

10.3 Problem data

We employ data gathered from several experiments performed at the Fort Leonard Wood Self Healing Minefield Test Facility ([156]). Those experiments deployed WSNs containing from 79 to 94 sensor nodes, which are custom design on an SH4 microprocessor running at 200 MHz. The nodes are equipped with four independent speakers and microphones and use ToA on the acoustic signal to determine the distance between themselves ([171]). The WSN was deployed on an area of $200\ m \times 50\ m$.

In total, there are 33 sets of distance measurements collected over the course of a few days. Each set consists of a single round of acoustic signal transmission by all the nodes. In a practical scenario, this kind of knowledge can be acquired in two ways. The first is to do as explained here: perform some previous experiments from which the measurement error model can be compiled. This is not always feasible, therefore a second, on-the-fly approach can alternatively be adopted. In this approach we assume some beacon nodes are within measurement range. In that scenario, the combination of GPS-known locations and measured distances can be used to establish the measurement error model.

10.3.1 Specific models for the used problem data

Figure 10.4 shows a graphical representation containing all the data from the 33 sets. With this data we can build up the models previously commented. In the figure, each dot represents a link whose distance has been measured; its abscissa value is the *measured* link distance, while its ordinate value is the *real* distance. If the two coincide, the measurement is correct, if they differ, the measurement is wrong. We notice that the majority of points tend to arrange themselves close to the diagonal, thus the majority of measurements are correct. However, for low or high distance values, the dots are spread in a cloud fashion, thus the corresponding measurements tend to be inaccurate.

Weighting function

Using these measurements, the weighting function for the L_1 norm function is calculated following Equation 10.3 with $margin = 50$ cm, and $NORM = 5$. Figure 10.5 illustrates the resulting weighting function. We can see how the function assigns higher weights to links with measured distances between 5 and 35 meters, which are thus the distance measurements with highest accuracy, there is a transition zone for distances from 35 to 45 meters, and measured distances below 5 or over 45 meters are heavily discriminated against with low corresponding weight values, since they are found to be the less reliable ones. This behavior closely matches the observed nature of measurements in Figure 10.4.

Likelihood function

The data displayed in Figure 10.4 is also used to generate the likelihood function: for this, a kernel function is centered at each of the represented points; as explained in Section 10.1, we employ a pyramidal smoothing kernel function with a base diagonal of $1\ m$. Figure 10.6 shows the resulting probability density functions obtained for five different values of measured link distance: 10, 20, 30, 40, and 50 meters. For simplicity we show all five PDF functions superimposed.

However, an analysis of this likelihood function showed that with the data available for this work, for almost any given candidate solution there is always some link producing a probability of zero (even for optimal solutions). This renders the likelihood function virtually useless, since a single zero turns the global product into zero. In order to avoid this, we establish a minimum probability floor, that ensures that improbable links will not produce a zero likelihood, but rather a very low value. After some experimentation, this ground value was set to 10^{-6}. Additionally, in order to cope with the enormous range of values of this guiding function, we use a logarithmic scale rather than the linear one.

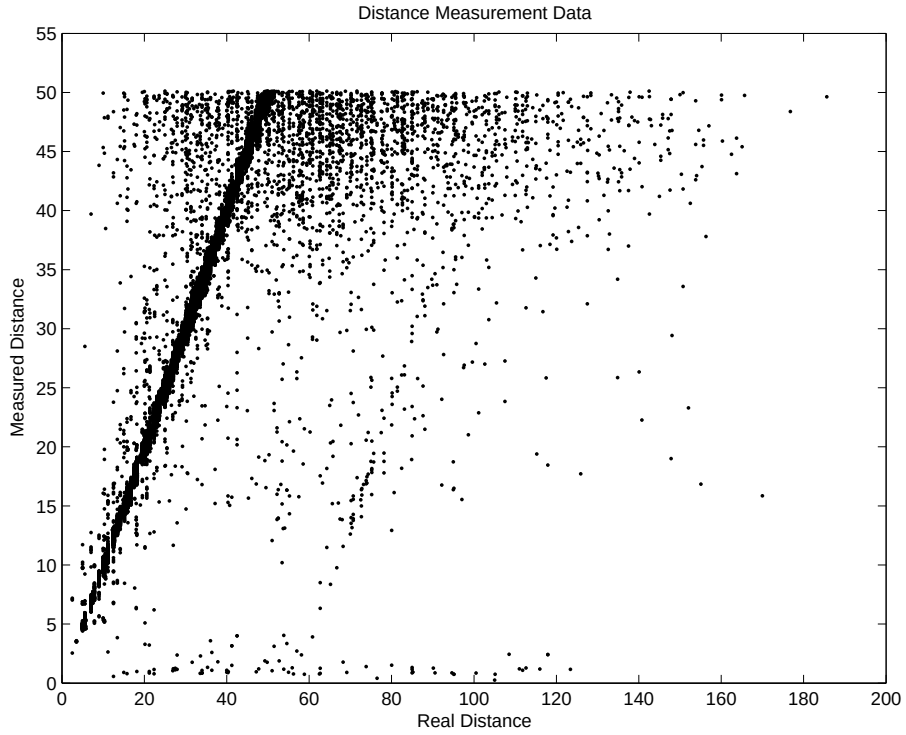


Figure 10.4: Distance measurements plot.

10.4 Two-Stage resolution process

In this section we present our novel proposed approach to solve LD. This approach is motivated by a study on the consistency of the most commonly used guiding functions in LD: the error norm and the Maximum Likelihood. We first present this study, later describe the two-stage approach, and finally discuss a beacon-reinforcement enhancement for the guiding functions in LD.

10.4.1 Guiding function consistency

It has already been said that the main objective of location discovery is to reduce the location error, that is, the distance between the real positions and the position estimations. However, we do not employ this parameter as the guiding criterion to our optimization technique, since it is unrealistic to assume we already know the real sensor nodes locations. Instead, a guiding function like the ones described above is employed to evaluate the solutions. It is only natural thus to ask oneself whether the guiding function selected is correctly leading the algorithm towards better solutions, that is, whether the evaluated value and the location error of a given solution are correlated.

In order to provide some insight onto this issue, we will use a simple yet effective criterion: let s_a and s_b be two possible solutions for a given LD instance, and $LE()$ the location error function; if $LE(s_a) < LE(s_b)$ then the guiding function should favor s_a over s_b. If this is the case, then we say the guiding function is *consistent* for this pair of nodes.

We define two different scenarios in order to apply our criterion. The first scenario consists of a pool of randomly generated pairs of solutions (low quality solutions). The second scenario consists of pairs of random solutions with low average location error (high quality solutions); for this scenario the solutions are

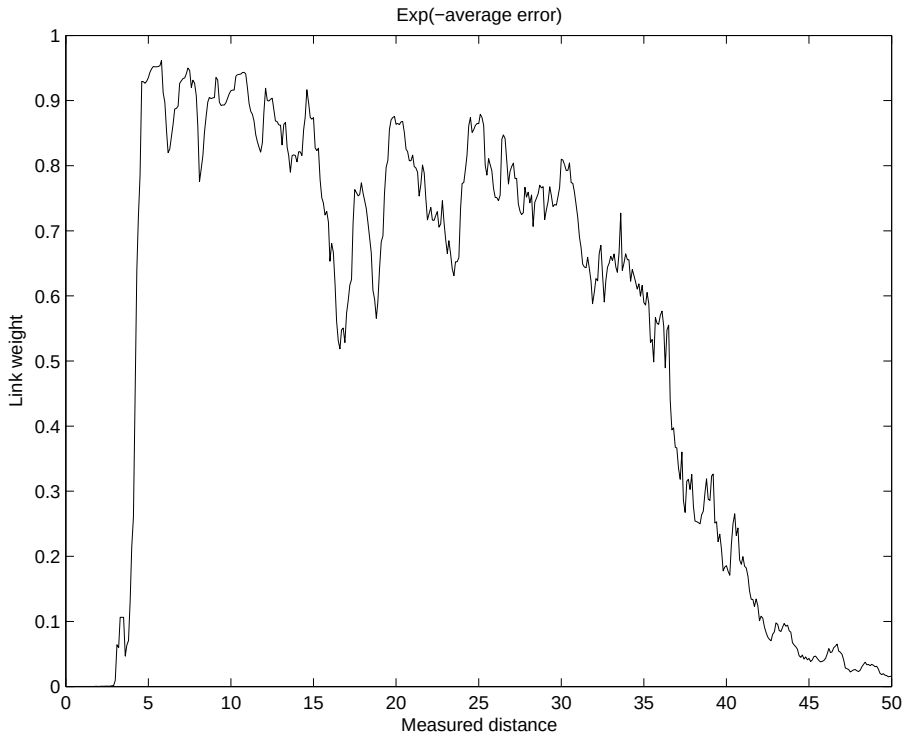


Figure 10.5: Weighting Function.

generated by adding low power white Gaussian noise to the real locations of the nodes.

Algorithm 10 shows the pseudocode of the consistency check performed; the *Initialize* function (lines 3 and 4) depends on the considered scenario, the *Evaluate* function (lines 5 and 6) is the corresponding guiding function (error norm or likelihood), and the evaluation values are considered *better* (line 8) when they are *higher* if the guiding function is the likelihood, or *lower* if it is an error norm. Finally, the returned value is the percentage of consistent solution pairs (line 13).

Algorithm 10 Guiding Function Consistency Check

1: $consistency = 0$
2: **for** 10000 **do**
3: Initialize(S_a)
4: Initialize(S_b)
5: Evaluate(S_a)
6: Evaluate(S_b)
7: **if** LocationError(S_a) $\leq$ LocationError(S_b) **then**
8: **if** S_a Has better fitness than S_b **then**
9: $consistency++$
10: **end if**
11: **end if**
12: **end for**
13: **return** $consistency/100$

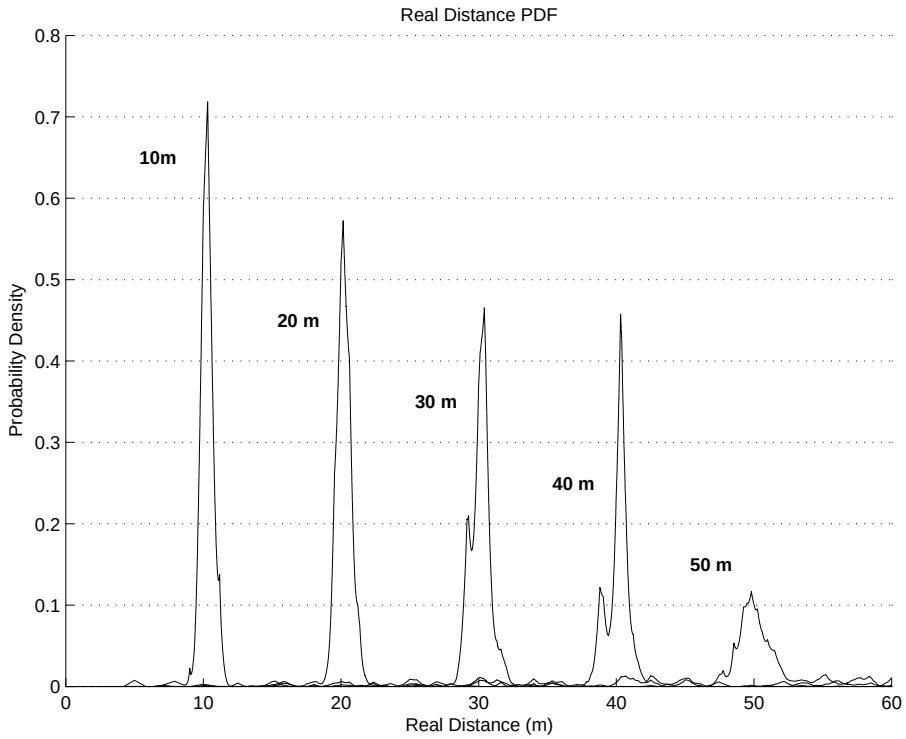


Figure 10.6: Kernel error model

For each scenario we generate 10000 random pairs of solutions, and compare the Maximum Likelihood function, the L_1 norm function, and the location error for the two solutions. For the second scenario we have used three different power levels for the noise: $0db$, $-10dB$ and $-20dB$ corresponding the average location errors of $1\ m$, $10\ cm$, and $1\ cm$, respectively. The percentage of pairs of solutions where each function is consistent (the solution with the lowest location error is evaluated as the best) are shown in Table 10.1; we also include the consistency of the L_∞ norm –which is not used in this work– as a reference.

Table 10.1: Consistency of the Maximum Likelihood (ML), L_1 and L_∞ norm functions for different location errors ($\%$).

Scenario	ML	L_1	L_∞
Pure random	55.8	62.1	54.7
WGN 100 cm	69.1	66.8	49.4
WGN 10 cm	68.5	55.8	50.4
WGN 1 cm	71.7	51.9	50.1

For the first scenario (pure random), L_1 gets higher consistency than ML: 62.1% vs. 55.8%. For the second scenario with $0dB$ error power (1 meter of average error), the consistencies are 68.9% and 66.3% for ML and norm guiding functions, respectively. If the noise power is reduced to $-10dB$ (10 cm on average) the consistency values become 68.0% and 56.1%, and for $-20dB$ (1cm) they become 71.6% and 51.6%, respectively.

From Table 10.1 it can be appreciated that ML acts only slightly better than random search (which would have 50% consistency) when the solution is far from the optimum. The same holds true for the L_1

norm function when the solution approaches the optimum. Therefore, we can state that the norm function is preferable when the solution is far away from the optimum, as in the beginning of the search process, and the ML function is preferable when the solution is close to the optimum, as in the end of the search process. As a result, we propose a new approach for solving this problem, that is described in the following section.

10.4.2 Two-stage Resolution

If we use the L_1 norm function alone, the obtained accuracy is expected to be limited, but it provides good guidance when the current solution is far away from the optimum, and the local optima are not extremely sharp. If we use the likelihood function, it provides a highly improved accuracy in the neighborhood of the optimum, but its guidance is poor in regions far away from it, and the local optima can be very strong. Therefore, using the L_1 norm seems a good idea when starting from a randomly generated solution, since it is likely to guide the search towards the neighborhood of the optimum; once the search process approaches that neighborhood, it is convenient to switch to a likelihood estimation, since it will produce much more accurate results in that narrow region.

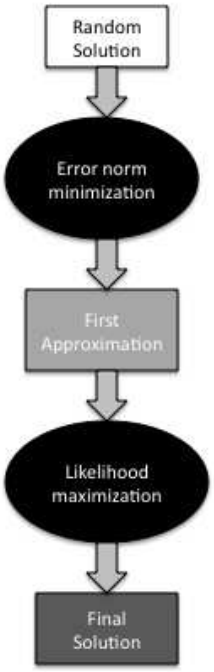


Figure 10.7: Two-Stage resolution process combining the first stage using error norm function and the second stage with a likelihood function.

Therefore, as in [49], we propose a two-stage solving process that combines two search processes. Figure 10.7 shows the basic configuration. The basic intuition is to use an initial phase to generate a rough initial guess by using L_1 starting from a random initial solution, then a second phase to refine the initial guess by using ML. The key feature for the first phase is robustness, we want to obtain a solution that has an upper bounded location error. The key feature for the second phase is accuracy, we want to minimize the location error as much as possible.

10.4.3 Beacon Reinforcement Factor

As was commented in Section 9.5, one of the major difficulties that arise during LD is the apparition of flips or rotations in a part of the WSN, generally by a cluster of nodes. This happens when a set of nearby nodes (the cluster) contains many references among nodes inside the set, and very few between nodes in the set and nodes outside of the set. As a result, the cluster is a "floating" entity, and a translation, rotation or flipping of the complete set produces only small variations of the guiding function; it is thus very difficult to be detected by the search algorithm, even when the location error suffers a large increase [163]. An example of flip/rotation error is shown in Figure 10.8, where real locations are indicated with dots, estimated locations with asterisks and a dotted line links every estimated location with its corresponding real location. We can see that almost every dotted line intersects at a single point: the rotation center.

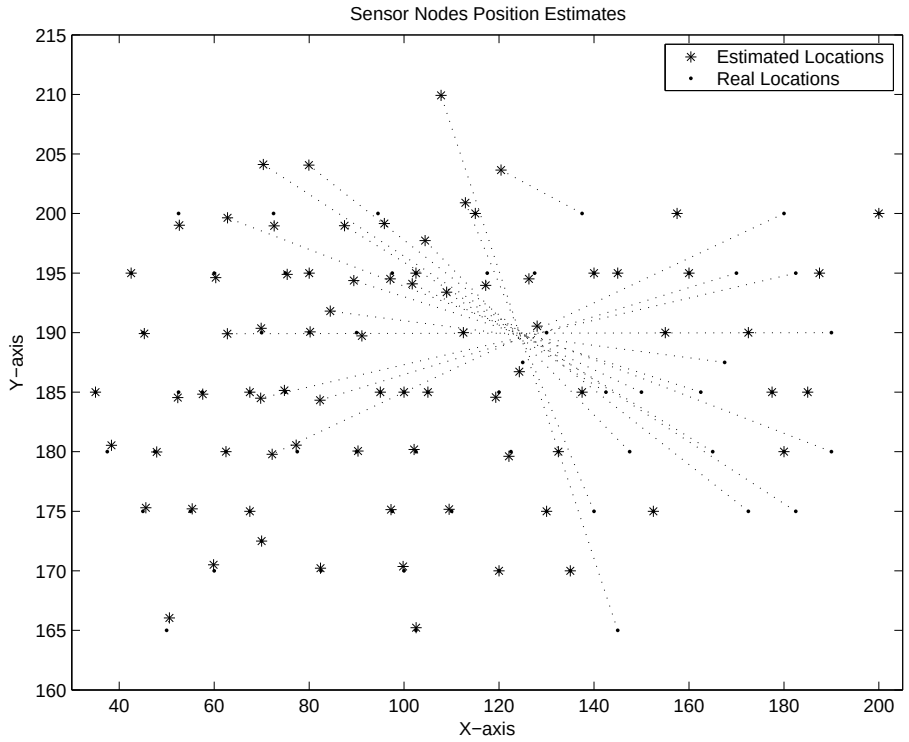


Figure 10.8: Example flip error: a cluster of nodes has their location estimations reflected through a point; this error is hard to detect when there are few distance measurements from nodes in the cluster to nodes outside the cluster.

Once the flip has occurred, it is very difficult for the search technique to fix it, since the flipped nodes produce an *attraction* effect on the remaining nodes of the cluster, stopping them from returning to their real locations. Speaking in optimization terms, a cluster displacement constitutes a local optima, and a very strong one for that matter. Therefore, there is a need for special mechanisms that helps escape this trap, or prevent falling into it in the first place.

There are some heuristic factors that can be incorporated to the guiding function and can help improve its performance. Since a beacon cannot be moved away from its location, this issue is usually solved when one or more nodes in the cluster set is a beacon. However, due to the reduced number of beacons, this is generally not the case. We propose to reinforce the effect of the already existing beacons in the network as a way to avoid translation, rotation, of flip errors. To do this, we assign a higher weight to those links

in which one of the nodes is a beacon node. This will force those nodes that have link distances measured with respect to some beacon to keep those distances, specially the beacon's close neighbors. For this work we have chosen a weight of 2 for the links containing a beacon; this weight is used both in the L_1 norm and the ML functions.

10.5 Problem instances

Of the 33 total data sets, we select 10 data sets to generate the problem instances, and the remaining 23 will serve as the data to establish the model. The main defining properties (number of nodes, number of link measurements available, and average measurement error per link) of the selected instances are summed up in Table 10.2. As can be seen, the average measurement errors are rather important, ranging from $1.70m$ to $4.58m$.

Table 10.2: LD problem instances features.

Instance	**3-19A**	**3-19B**	**3-19C**	**3-19D**	**3-19E**	**3-19F**	**3-20A**	**3-20B**	**3-25A**	**3-25B**
Number of nodes	79	93	93	94	94	94	94	93	93	94
Number of links	677	673	394	644	378	622	978	1026	992	1279
Avg. link error (m)	4.55	3.89	2.05	2.99	1.70	2.52	3.51	2.92	2.55	4.58

In all of our instances, we set a small number of the nodes to become *beacons* and serve as reference points for the rest of nodes to locate themselves. The default quantity of beacon nodes in our LD formulation is 10% of the nodes in the network. The beacons are chosen randomly among the nodes in the deployed WSN; in order to avoid possible dependencies with the chosen beacons, for each problem instance we select ten random beacon configurations, labeled from '1' to '10', and solve each beacon configuration of each problem instance as an instance.

In all the data sets, we have complete knowledge about the real locations of the nodes (otherwise neither Figure 10.4 nor the data-dependent models could have been generated), however the optimization algorithms have no access to this information. We use this information in a post-processing estimation of the nodes location accuracy, as a mean of evaluation of the LD processes.

10.6 Experiments

In this section we describe the experimental tests conducted on the LD problem. We pick SA as the base technique to solve LD, and the L_1 norm function as its base configuration for comparison purposes. We test the effect of the different configurations against it, and the sensibility of the obtained solutions with respect to the beacon density in the WSN. Later, we compare the performance of the different optimization algorithms: SA, GA, and PSO. Finally, we assess the effectiveness of our proposed two-stage resolution process by comparing its results to the results produced under the same conditions using either the L_1 error norm function or the likelihood approach.

The parametric configurations of the algorithms were empirically tuned; the values obtained are shown in Table 10.3. All the solutions are obtained after performing 100 independent executions of $5,000,000$ evaluations each (since there are ten instances with ten beacon configurations each, in total $10,000$ independent executions are performed for each test scenario). For the comparisons, we show the boxplots of the average location errors for the non-beacon nodes, for each of the ten problem instances. In each graphical representation, the same scale is used for all the representations, to enable visual comparisons among plots.

Table 10.3: Parametric configurations of the optimization algorithms.

Algorithm	**GA**
evaluations	$5,000,000$
population	100
selection	*Roulette*
replacement	*8-Tournament*
crossover	0.80
mutation	$\begin{cases} p_m = 1/L \\ R_{max} = 15 \end{cases}$

Algorithm	**PSO**
evaluations	$5,000,000$
swarm	50
$C1$	2
$C2$	2
starting inertia	0.5
final inertia	0.1

Algorithm	**SA**
evaluations	$5,000,000$
mutation	$\begin{cases} p_m = 1/L \\ R_{max} = 15 \end{cases}$
Markov chain	50
α	0.99995

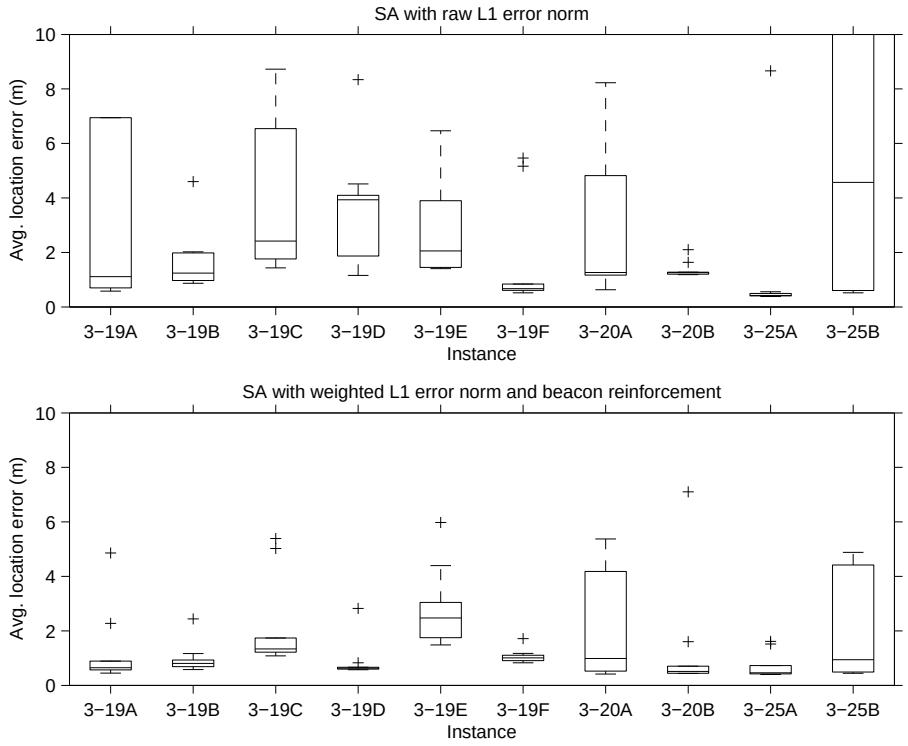


Figure 10.9: Effect of link weighting and beacon reinforcement in the L_1 error norm.

10.6.1 Impact of the Link Weighting and the Beacon Reinforcement

We start by analyzing the effects produced by the use of link weighting and beacon reinforcement over the basic L_1 error norm function. For this, we run SA using the raw error norm function as its guiding function, and SA using the error norm function with both link weighting and beacon reinforcement (see

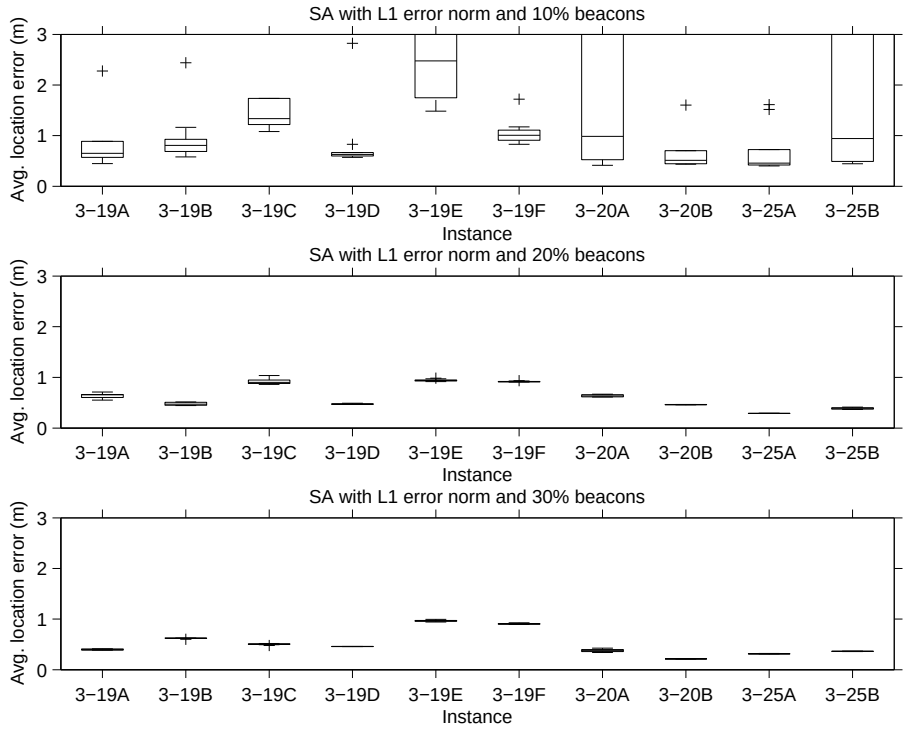


Figure 10.10: Influence of the beacon density.

sections 10.3.1 and 10.4.3). The location errors obtained for the ten selected problem instances are displayed in Figure 10.9. For all scenarios, the parametric configurations are like the one presented in Table 10.3.

Figure 10.9 shows the boxplot representation of the global average location errors obtained with the raw L_1 error norm function for the ten problem instances (top), and the average location errors obtained with L_1 error norm function using link weight and beacon reinforcement (bottom). The configuration of SA that uses raw error norm function performs noticeably poorer than the base configuration of SA: for instances 3-19A, 3-19C, 3-19D and 3-25B the location errors are visibly larger, while for the rest of the instances both configurations perform similarly.

Therefore, we conclude that adding link weight and beacon reinforcement improves the performance of an optimization algorithm that uses the L_1 error norm function, since in several instances it achieves location errors lower by an order of magnitude, while for the rest of instances the achieved location errors are similar or slightly lower.

10.6.2 Influence of the beacon density

In this section we study how the density of beacons in the WSN affects the overall location error. Our intuition says that for higher beacon densities, the expected resulting location error should become smaller. The base technique is SA with L_1 error norm, link weight, and beacon reinforcement. Figure 10.10 shows the boxplot representation of the average location errors for WSN with: 10% beacon nodes (top), 20% beacon nodes (center), and 30% beacon nodes (bottom). Note that the beacon configurations in the two additional scenarios defined cannot match the ones used in the test case (since different numbers of beacons are used).

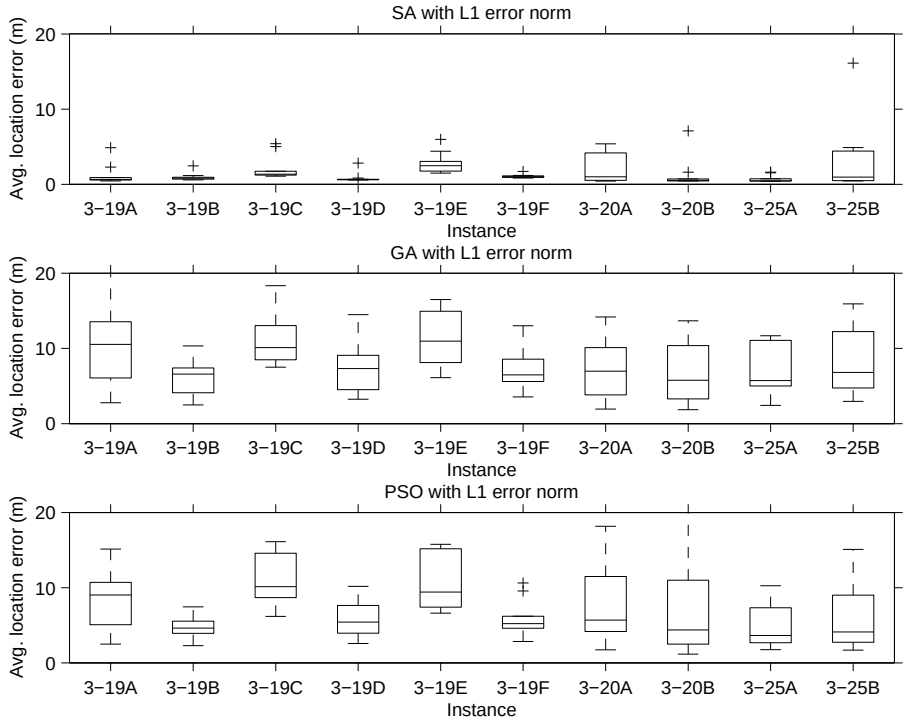


Figure 10.11: Performance of the different optimization algorithms.

The boxplot representations match the aforementioned intuition concerning the influence of the beacon density. As can be seen, the solutions produced with 20% beacon nodes achieve lower location errors in all ten instances, the largest improvements being found in instances 3-19E, 3-20A and 3-25B. Additionally, the achieved location errors show much smaller variances. These properties are even more pronounced when the node density is augmented to 30%, with the lowest average location errors and almost zero variance; the differences between using 20% and 30% beacon nodes are small, however, in fact the average location error in the first case ranges from 0.29m to 1.04m, while in the second it ranges from 0.21m to 0.99m, depending on the problem instance considered.

The statistical analysis confirms that the location errors of solutions of instances containing either 20% or 30% beacon nodes are significantly lower than those with 10% beacon nodes, for any of the problem instances. However, switching from 20% to 30% beacon nodes does not bring a clear improvement: the error is significantly lower in 6 problem instances, but in the remaining 4 instances it is significantly higher. Therefore, we recommend 20% beacon nodes as the optimal trade-off value between price and accuracy.

Despite these results, the test instances we use for the rest of experiments contain 10% beacon nodes. The reason for this is to test the solving techniques in the most challenging yet feasible scenario. From the results of this section, the performances of the optimization techniques are expected to significantly improve if the beacon density is increased to 20%, for any of the problem instances at hand.

10.6.3 Performance of the different algorithms

In this section we test the relative performances of the different optimization algorithms selected for LD. Their parametric configurations were empirically tuned, and are displayed in Table 10.3. The base test instances are the same used so far, with 10% beacon nodes randomly selected in ten different configurations

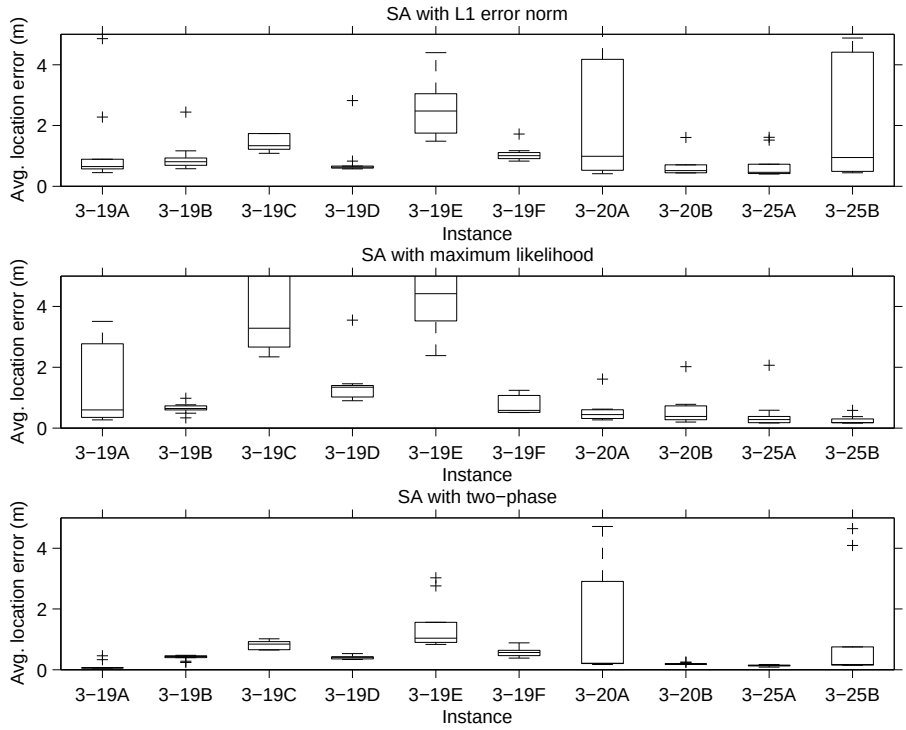


Figure 10.12: Results of the different search processes.

per instance (the same sets of beacon configurations are used in all the algorithms). The three algorithms use the L_1 error norm function with link weighting and beacon reinforcement as the guiding function. The boxplots of the average location errors produced by the algorithms is shown in Figure 10.11.

The results displayed clearly show that SA outperforms both GA or PSO in the LD problem. In effect, for all of the ten instances at hand, the average location errors produced by SA are visibly lower (and have lower variance) that those obtained by the other two techniques. Between GA and PSO the differences are slight: for instances 3-19B and 3-19F PSO obtains lower errors, but for the remaining eight both algorithms perform similarly. Additionally, the statistical analysis of the results performed for the 100 scenarios (combination of ten problem instances and ten beacon configurations) points out that the location error of the solutions obtained using SA are significantly lower than those obtained with GA in 95 cases, and significantly lower than those obtained with PSO in 94 scenarios. As a result, we state that SA is the best performing algorithm for LD, outperforming both GA and PSO.

10.6.4 Comparison of the different search processes

Finally, we are ready to assess the effectiveness of our proposed two-stage search technique. For this, we select SA as the optimization technique, and establish the comparison for three different configurations: L_1 error norm as the guiding function (with both link weight and beacon reinforcement), likelihood guiding function, and our proposed two-stage approach (Section 10.4.2) using L_1 error norm in the first stage, and the likelihood function in the second stage. The results obtained in this experiment are displayed in Figure 10.12. The balance between the two phases has been empirically determined. The chosen configuration is $4,000,000$ solution evaluations in phase 1 (L_1 error norm) and $1,000,000$ solution evaluations in phase 2. This configuration is the result of the LD problem complexity, in which the task of finding an approxi-

mate solution is relatively more complex than refining that solution. Note that the combined computational effort of the two phases equals the computational effort of each of the other two approaches, for which the stopping criterion is $5,000,000$ solution evaluations.

Between the L_1 error norm and the likelihood function there is no clear winner; for instances 3-19C and 3-19D the L_1 error norm obtains lower location errors, while for instances 3-20A and 3-25B the opposite happens. Finally, our proposed two-stage approach receives the best part out of each of them, and outperforms both in all problem instances (except for 3-20A, where it is slightly outperformed by the likelihood function). The results from the statistical analysis state that the two-stage search process is significantly better than the test case (L_1 error norm) in 95% of the test scenarios, and worse in only 5%, it is better than the likelihood in 94%, and worse in only 5%. Therefore, we conclude that the two-stage search process produces a real improvement over single phase solving processes, obtaining significantly lower location errors in over 94% of the tested cases.

10.7 Conclusions

In this chapter we have addressed the resolution of the LD problem in WSNs. In this problem a set of nodes is deployed in a terrain and nodes take certain inter-node distance measurements. Among the nodes, a small subset has self-locating capabilities, and their locations are used are absolute references by the rest of the nodes.

In our formulation, the objective of LD is to obtain the coordinates of the nodes such that the average node location error is minimal. However, the location error cannot be computed in a real scenario, so the fitness or guiding function has to be defined otherwise; we consider two alternatives for it: error norm functions, and Maximum Likelihood. We incorporate a link weight factor to discriminate links according to the reliability of the distance measurement they have in the error norm function, and a beacon reinforcement factor that doubles the contribution of each link in which one of the nodes is a beacon node, in order to avoid flip errors.

After conducting a study of the consistency of the two guiding functions available, we conclude that each one outperforms the other under different circumstances: for solutions with large errors, the error norm performs better, while for solutions with low errors, the likelihood performs better. Therefore, we propose a two-stage approach that uses error norm first, then switches to likelihood when the solution is considered to contain lower error.

We define a test bench of 10 instances selected from 33 sets of real distance measurements; the 23 remaining sets serve to construct the models used for link weighting and likelihood. For each selected instance we generate 10 random beacon node configurations, thus obtaining a total of 100 test scenarios. We conduct the first series of experiments to assess the effectiveness of the link weighting and beacon reinforcement factors, the influence of the beacon node density, and the performance of the different algorithms. The results show that link weighting and beacon reinforcement noticeably contribute to reduce the average location error of the L_1 error norm function: in 50% of the instances adding link weighting and beacon reinforcement factor has reduced the errors by an order of magnitude of the errors, while for the remaining 50% the produced results have been of similar quality. We also show that 20% beacon nodes holds the optimal trade-off between accuracy and cost (we still keep the value to 10% to have a challenging problem). Regarding the algorithms, we highlight that SA largely outperforms GA and PSO, in 95% and 94% of the test cases, respectively.

Finally, we conduct the last set of experiments to test the effectiveness of the proposed two-stage solving procedure against both L_1 and likelihood approaches in isolation. The two base approaches outperform one another in different scenarios, however the proposed two-stage approach showed a clear improvement towards each of the other two, being able to equal or outperform both in 94% of the 100 test scenarios.

Part V

CONCLUSIONS AND FUTURE LINES OF RESEARCH

Chapter 11

Conclusions

This thesis work has tackled the resolution of complex optimization problems found in the domain of Wireless Sensor Networks (WSNs). This relatively new field has brought new and exciting possibilities for experimental sciences and industry, but also novel problems, and new hard constraints that must be dealt with. This combination of factors demands that new and powerful optimization techniques have to be developed and tuned in order to properly address them.

We have first made a review of the basic principles found in WNSs, including the models most commonly used for both the sensor nodes and the network itself. We have listed the special features that distinguish WSNs from other networks, specially regular ad hoc networks. We have provided short reviews of existing hardware platforms for sensor nodes, types of sensors, and current sensor network application fields. Then, we have provided a review of the main problems (with optimization component) that can be found in WSNs, with special attention to those that receive most attention from the research community.

We propose the use of metaheuristics as the key tool to address the resolution of the optimization problems chosen for this thesis, so we first offer a description of this kind of technique. We classify the techniques into the main categories found, depending on the way they handle the candidate solutions (population and trajectory), and the type of problem approach they solve (mono-objective or multi-objective). Additionally, we have explain the convergence model for distributed populations developed by Gabriel Luque since it serves as the basis for our automatic migration tuning technique.

Regarding the problems, we have picked two of the most addressed ones found in the literature, the layout optimization problem (WSNL) and the location discovery problem (LD), and additionally, we have addressed a third problem, the radio network optimization problem (RND), that is closely related to another problem found in WSNs, the sensor node scheduling problem. We now describe them shortly in turn. In RND, the task consists in selecting a subset of locations from a set of available locations for the installation of base stations (BTs), the objectives are to maximize the radio coverage these BTs will provide and minimize the total number of locations selected. In the WSNL problem, the task consists in deciding the number of sensor nodes and the geographic locations for their deployment as a WSN, with the objectives of maximizing the sensing coverage, minimizing the economic cost (expressed through the number of sensor nodes), and maximizing the lifetime of the system (by reducing the energy consumption due to communications). In LD, the task consists in finding the geographic locations of the nodes of a deployed WSN from a set of node-to-node distance measurements and landmarks (or beacons), the objective is to minimize the average node positioning error.

These problems address different concepts found in WSNs; both RND and WSNL are *design* problems, where the network or some aspect of it has to be designed in order for the resulting system to meet some quality standards, meanwhile the LD problem is an *analysis* problem, where some properties of the network (in this case, the location of nodes) have to be approximated. It should be noted that our approaches to these problems are not limited to abstract academic definitions; we develop complex realistic problem models

and instances (for RND and WSNL), or even solve real problem instances when possible (for LD). From a resolution point of view, the approaches to these problems also cover a wide spectrum, since LD is a mono-objective problem, WSNL is a multi-objective problem, and RND has been defined under both mono and multi-objective approaches. We now proceed to describe the work done for each of the three problems in detail.

In the RND problem, we propose CHC as a competitive solving algorithm in the mono-objective problem formulation, since it consistently outperformed (100% of the test cases) both SA and GA in 10 problem instances of different dimensions and geometry. We developed a multi-objective version of that algorithm, MOCHC, and showed that it is also highly competitive by comparing it against the state-of-the-art algorithm NSGA-II in the same set of instances and noticing that it obtained better results (again, in 100% of the test cases). Both CHC and MOCHC were successfully expanded to deal with non-binary solutions as the problem was developed to include directional antennae, for which a direction parameter needs to be defined. Finally, in a wide spectrum comparison against 13 state-of-the-art optimization algorithms conducted on the large real-world based instance of Malaga, performed in cooperation with other research groups, CHC ranked third. In addition, using that same instance of Malaga as a test bench, we tested a novel theory-driven proposal to automatically configure the migration parameters in a parallel Genetic Algorithm. We managed to obtain results similar to the best performing fixed migration schedules found during an empirical parameter tuning, and only slightly worse than the best found sequential configuration, while avoiding the burden of parameter tuning; the overall computational effort was estimated to achieve savings of over 75%.

In the WSNL problem, we propose a new local improvement operator, PACO, to be used integrated with an optimization algorithm. Our proposed operator searches the candidate solution for local inefficiencies due to two nodes being too close to one another, then tries to fix it by replacing the close nodes by a single node capable of maintaining the network's original coverage and connectivity. The effectiveness of the operator is proved for four state-of-the-art multi-objective metaheuristics: NSGA-II, PAES, SPEA2 and MOCell. Two kinds of genetic operator we tried for both the mutation and the crossover, a 'random' operator on the one side (SBX for the crossover), and a 'geographic' operator on the other side. The use of PACO improved the results in the wide picture with a probability of 84.85%, but when the study is restricted to the best performing half set of algorithmic configurations, then PACO brings improvement in 98.48% of the cases. Additionally, the performance of PACO has been found to improve when the dimension of the problem instance grows, making it even more attractive to the domain of WSNs, since networks are expected to contain high numbers of nodes in the near future.

In the LD problem, we study the use and consistency of the two most popular kinds of guiding function: the error norm functions and the likelihood functions. After the study, we conclude that each one outperforms the other under different circumstances, and thus propose a new two-stage resolution method to take advantage of each of the functions strengths. To test our proposal, we use a set of 10 *real* problem instances, and additional data from another 23 measurement data sets to establish the required models. SA, GA, and PSO are the algorithms chosen for our experiments. We found that SA produced better results than the two others, that link weighting and beacon reinforcement greatly improve the performance of the error norm function, and that by using 20% beacon nodes the expected accuracy is close to optimal. Finally, the effectiveness of the proposed two-stage approach was proved in SA by comparing its results to those of either guiding function working separately; the best average location error obtained by using only one guiding function is improved by the two-stage approach in 94% of the test cases.

As a general evaluation of the thesis work, we have tackled three of the most important problems found in (or closely related to) the domain of WSNs, and have solved all of them satisfactorily using metaheuristics. Additionally, we have proposed a novel contribution in each of the problems that help improve the solutions obtained, or that help reduce the required time and computational effort to solve the problems. Each of these contributions explores an important concept found in optimization. Parallelism is explored in the RND problem, where we have proposed an automatic migration tuning technique for dGA, thanks to which high-quality solutions are obtained in shorter times. Integration of problem knowledge in specific advanced operators is explored in the WSNL problem, where we have proposed a novel local

improvement operator that, used in combination with a metaheuristic algorithm, helps the latter improve the quality of the solutions produced. Finally, the combination of different search techniques is explored in LD, where we have proposed a two-stage combination of two types of guiding function, error norm and likelihood, that produces results with lower location errors than either of the former separately. Another concept that has a noticeable importance throughout this work is multi-objective optimization; in WSNL it constitutes the approach chosen for the problem (with additional constraints), and in RND its use is validated by comparing its performance against that of mono-objective techniques under equivalent circumstances.

This thesis has produced significant publications of high impact related to its different contributions. The following references are published papers in ISI listed journals, and can be found in Appendix A in the following. The performance of CHC for solving RND was stated in [3]. The effectiveness of metaheuristics applied to the WSNL problem was presented in [4], and the benefits of using PACO in that problem were shown in [1]. Finally, the power of the combination of different search techniques for LD was demostrated in [2]. Furthermore, the results of the work developed in this thesis have direct applicability to real-world problems like the site selection problem for cellular networks (taken from RND), or the geographic location of sensor nodes or mobile devices (taken from LD); strategies for the deployment of sensor nodes can also be developed from our proposed methods in WSNL.

As future lines of research, we can identify two main trends. The first one is problem-oriented, and consists in developing new, more complex, and more accurate problem instances and models. The second is technique-oriented: to add a strong focus towards distributed execution of the optimization techniques. Possible first steps in this second trend are to develop geographic partitions of the network and solve the problem locally for each subnetwork, then integrate the solutions.

Part VI

APPENDICES

Appendix A

List of publications related to this thesis work

In this appendix we present the set of works that have been published during the years in which this thesis work has been developed. These publications speak for the interest, validity, and impact on the scientific community and literature of the work contained in this thesis, since they have appeared in prestigious forums, and have been subject to peer review by expert researchers. Figure A.1 shows a diagram of the different publications, and their relationships with the contents of the work. We list these publications next.

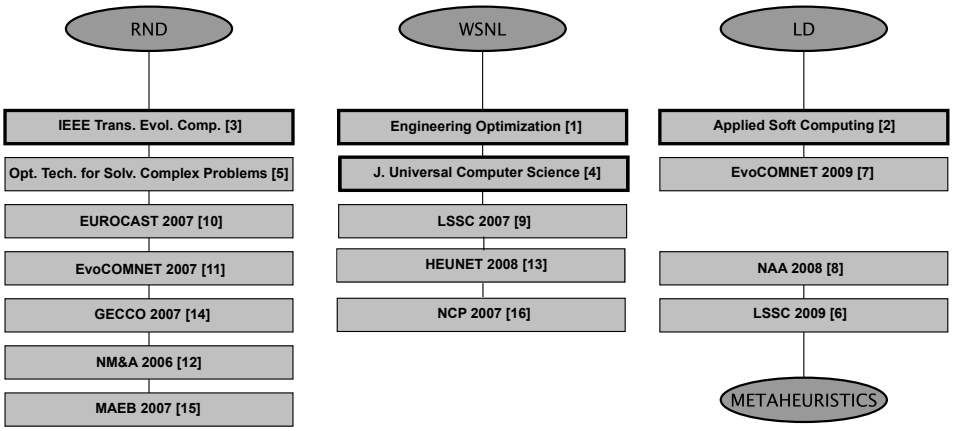


Figure A.1: Diagram of the publications related to this thesis work.

ISI JCR indexed journals:

[1] G. Molina, F. Luna, A.J. Nebro, and E. Alba. An efficient local improvement operator for the multi-objective wireless sensor network deployment problem. *Engineering Optimization*, *Accepted for publication*, 2010.

[2] G. Molina and E. Alba. Location Discovery in Wireless Sensor Networks Using Metaheuristics. *Applied Soft Computing*, in press, corrected proof, 2010.

[3] S. Priem-Mendes, G. Molina, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, Y. Sáez, G. Miranda, C. Segura, E. Alba, P. Isasi, C. León and J. M. Sánchez-Pérez. Benchmarking a Wide Spectrum of Meta-Heuristic Techniques for the Radio Network Design Problem. *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pages 1133 – 1150, 2009.

[4] G. Molina, E. Alba, and E-G. Talbi. Optimal Sensor Network Layout Using Multi-Ob jective Metaheuristics. *Journal of Universal Computer Science*, vol. 14, no. 15, pages 2549 – 2565, 2008.

Book chapters:

[5] G. Molina, J. F. Chicano and E. Alba. Optimal Location of Antennae in Telecommunication Networks. In *Optimization Techniques for Solving Complex Problems*, Wiley (en prensa), 2008.

International conferences of the series *Lecture Notes in Computer Science*:

[6] S. Fidanova, E. Alba and G. Molina. Hybrid ACO Algorithm for the GPS Surveying Problem. In *Proceedings of the Large-Scale Scientific Computations (LSSC 09)*, volume 5910 of *LNCS* pages 318–325, 2009.

[7] G. Molina and E. Alba. Location Discovery in Wireless Sensor Networks Using a Two-Stage Simulated Annealing. In *Applications of Evolutionary Computing: EvoCOMNET 09*, volume 5484 of *LNCS*, pages 11 – 20, 2009.

[8] S. Fidanova, E. Alba and G. Molina. Memetic Simulated Annealing for the GPS Surveying Problem. In *Fourth International Conference on Numerical Analysis and its Applications (NAA 08)*,volume 5434 of *LNCS* pages 281–288, 2008.

[9] E. Alba and G. Molina. Optimal Wireless Sensor Network Layout with Metaheuristics: Solving a Large Scale Instance. In *Proceedings of the Large-Scale Scientific Computations (LSSC 07)*, volume 4818 of *LNCS*, pages 527 – 535, 2007.

[10] M. A. Vega-Rodr´guez, J. A. Gómez-Pulido, E. Alba, D. Vega-Pérez, S. Priem-Mendes, and G. Molina. Using Omnidirectional BTS and Different Evolutionary Approaches to Solve the RND Problem. In, *Eleventh International Conference on Computer Aided Systems Theory (EUROCAST 07)*, volume 4739 of *LNCS*, pages 853 – 860, 2007.

[11] M. A. Vega-Rodr´guez, J. A. Gómez-Pulido, E. Alba, D. Vega-Pérez, S. Priem-Mendes, and G. Molina. Evaluation of Different Metaheuristics Solving the RND Problem. In *Applications of Evolutionary Computing: EvoCOMNET 07*, volume 4448 of *LNCS*, pages 101 – 110, 2007.

[12] E. Alba, G. Molina and J. F. Chicano. Optimal Placement of Antennae using Metaheuristics. In *Numerical Methods and Applications (NM&A 06)*,volume 4310 of *LNCS* pages 214–222, 2006.

Other national and international conferences:

[13] G. Molina and E. Alba. Wireless Sensor Network Deployment Using a Memetic Simulated Annealing. In *InternationalSymposiumonApplicationsandtheInternet (HEUNET 08)*, pages 237 – 240, 2008.

[14] A. J. Nebro, E. Alba, G. Molina, J. F. Chicano, F. Luna, and J. J. Durillo. Optimal antenna placement using a new multi-objective CHC algorithm. In *Genetic and Evolutionary Computation Conference (GECCO 07)*, pages 876 – 883, 2007.

[15] E. Alba, G. Molina, and A. J. Nebro. Disposición óptima de antenas usando CHC multiobjectivo. In *V Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB 07)*, pages 199 – 205, 2007.

[16] G. Molina, E. Alba. Optimal Location of Sensor Nodes with a Memetic Simulated Annealing. In *International Conference on Nonconvex Programming (NCP)*, 2007.

Appendix B

Resumen en español

Los avances recientes en la miniaturización de la electrónica han propiciado la aparición de dispositivos integrados de tamaño reducido con capacidades de cómputo, comunicación, y medición: los nodos sensores. Estos nodos forman el elemento constitutivo básico de las *redes de sensores*, un nuevo paradigma en el campo de las ciencias experimentales que ofrece prestaciones de medición y monitorización nunca antes vistas. Sin embargo, a la par que este campo abre nuevas posibilidades, un nuevo conjunto de problemas de gran complejidad ha de ser resuelto para lograr un comportamiento satisfactorio.

Ya existe un importante cuerpo de conocimiento sobre las redes de sensores, y en particular sobre el abordaje y la resolución de muchos de los problemas que en ellas surjen. Sin embargo, el rápido desarrollo experimentado por esta propuesta tecnológica hace que dicho cuerpo de conocimiento crezca, evolucione, y se modifique continuamente y cada vez con mayor ritmo, lo cual vuelve muchas de las anteriores propuestas obsoletas. Es por tanto necesario el proponer una serie de estrategias y métodos que permitan, de manera robusta, flexible, y eficiente, resolver distintos problemas cuyos planteamientos, por su naturaleza y complejidad, varía con frecuencia.

En este trabajo de tesis doctoral proponemos el uso de técnicas metaheurísticas para la resolución de algunos de los principales problemas que se hallan en las redes de sensores. Adicionalmente, por cada uno de los problemas considerados, proponemos una contribución novedosa que permite mejorar las prestaciones de la técnica resolutiva, ya sea en lo referente a la *calidad* de las soluciones obtenidas, o en la *eficiencia* del método, cuya efectividad evaluamos de forma experimental, con confianza estadística, sobre instancias complejas o incluso realistas del problema en cuestión.

B.1 Organización

Esta tesis doctoral se compone de cinco grandes bloques. En el primer bloque se presentan los fundamentos en los que se basa el trabajo: por una parte las redes de sensores, y por otro las técnicas metaheurísticas que sirven como base para la resolución. Las tres siguientes partes se ocupan de cada uno de los tres grandes problemas resueltos en este trabajo: el diseño de la red de radio (RND), el despliegue de nodos sensores (WSNL), y el descubrimiento de localización (LD). Finalmente, en el último bloque se agrupan los principales logros alcanzados en esta tesis y se extraen las conclusiones, tanto a nivel de conjunto como particularizadas por escenario. A continuación detallamos de manera específica el contenido por capítulos.

- **Capítulo 1: Introducción.** En este capítulo se realiza una justificación de las razones que motivan la presente tesis, y se esboza un esquema del contenido de la misma.
- **Capítulo 2: Redes de sensores.** Este capítulo describe de manera general los nodos sensores y las redes de sensores. Los modelos empleados para ambas entitades son presentados, así como las

principales características que distinguen a este tipo de red de otras redes a priori similares, como las redes inalámbricas ad hoc. También se presenta un breve listado de plataformas existentes, así como ejemplos de aplicaciones de uso. Finalmente, se realiza una revisión general de los problemas de optimización encontrados en relación con el uso de estas redes.

- **Capítulo 3: Metaheurísticas.** Este capítulo realiza una introducción genérica al campo de las técnicas metaheurísticas, incluyendo los principales conceptos que se emplean, e indicando las principales categorías en las que se clasifican. Se hace especial hincapié en los paradigmas específicos empleados en este trabajo: resolución de problemas multiobjetivo y metaheurísticas paralelas, incluyendo una breve descripción de un estudio terico sobre la convergencia en poblaciones distribuidas, ya que sirve como punto de partida para una de las contribuciones que vienen se exponen en los captulos siguientes.

- **Capítulo 4: Algoritmos.** En este capítulo se describe de manera general (como plantillas) los algoritmos que se utilizan para resolver los distintos problemas abordados.

- **Capítulo 5: Problema de diseño de la red de radio.** En este capítulo se describe el problema de diseño de la red de radio (RND). Los principales modelos empleados para la evaluación de la cobertura ofrecida por la red son presentados, y la literatura existente relativa a la resolución de este problema se revisa. A continuación se presenta en problema de planificación en redes de sensores, que guarda relación con RND, y dicha relación es explicada. Se propone una extensión de la resolución de RND para cubrir el problema de planificacin, y se revisa la literatura existente relativa a este ltimo problema.

- **Capítulo 6: Resolución del problema de diseño de la red de radio.** En este capítulo se describen las dos formulaciones empleadas en el planteamiento de objetivos del problema RND, monoobjetivo y multiobjetivo, así como los dos tipos de problema, binario (sin parámetros) y entero (con parámetros). Se definen ocho instancias de problema de distinta complejidad, y se resuelven con distintos algoritmos. Se propone un método de autoajuste de la migración basado en el estudio de convergencia presentado en el capítulo 3. Finalmente, empleando como base la mayor instancia definida, se demuestra la efectividad de la técnica de autoajuste de la migración.

- **Capítulo 7: Problema de despliegue de nodos sensores.** En este capítulo se describe el problema de despliegue de nodos de la red de sensores (WSNL). Se describe los distintos modelos existentes para la cobertura y las comunicaciones que se emplean en la literatura, tanto a nivel de nodo como a nivel de red. Se define el concepto de tiempo de vida, y se presentan los modelos más comúnmente empleados para su estimación. Finalmente, se realiza una revisión de la literatura existente para este problema.

- **Capítulo 8: Resolución del problema de despliegue de nodos sensores.** En este capítulo se describe la formulación multiobjetivo empleada para este problema, con el número de nodos y el tiempo de vida como objetivos, y la cobertura planteada como restricción. Se propone un novedoso operador de mejora local para las soluciones, PACO, para su uso integrado dentro de un algoritmo de optimización. La efectividad del operador propuesto se testea sobre instancias de distinto tamaño, y sobre 4 algoritmos del estado del arte, y dos conjuntos de operadores, aleatorios y geográficos.

- **Capítulo 9: Problema de descubrimiento de localización.** En este capítulo se describe el problema del descubrimiento de localización (LD). Se comienza con un breve repaso de las técnicas existentes para la medición de distancias entre nodos, sus limitaciones y errores, y los primeros métodos empleados para determinar las posiciones de los nodos. Posteriormente, se introducen los dos tipos de función más empleados para guiar las técnicas de búsqueda: norma del error y probabilidad. Finalmente, realizamos una revisión de la literatura relativa a este problema.

- **Capítulo 10: Resolución del problema de descubrimiento de localización.** En este capítulo describimos la formulación empleada para el problema LD. Se realiza un estudio de la consistencia de los dos tipos de funcion utilizados para guiar los mecanismos de búsqueda, y se propone un método de dos fases como resultado. Se comprueba la efectividad del método propuesto mediante la resolución de 10 instancias generadas a partir de un conjunto de datos experimentales reales.

- **Capítulo 11: Conclusiones** En este capítulo se resumen las principales conclusiones extraídas del trabajo realizado, tanto a nivel de conjunto como particularizadas por cada uno de los problemas abordados.

- **Apéndice A: Publicaciones.** En este apéndice se listan las publicaciones realizadas como consecuencia del trabajo enmarcado dentro de la presente tesis doctoral, así como su relación con los distintos contenidos de la misma.

- **Apéndice B: Resumen en español.** El presente resumen de la tesis.

B.2 Redes de sensores

Las redes de sensores (WSNs) son un nuevo tipo de redes sin infraestructura (ad hoc) compuestas por pequeños dispositivos llamados nodos sensores, cuyo propósito es medir o monitorizar una o más variables físicas dentro de un determinado entorno. Cada nodo sensor posee capacidad de cómputo y de comunicación (inalámbrica), y además incluye uno o más sensores (de ahí su nombre). Algunos tipos de sensor pueden ser: sensores mecánicos, sensores magnéticos y electromagnéticos, térmicos, ópticos, químicos, o acústicos.

Las principales características definitorias de los nodos sensores son, además de las ya expuestas, su reducido tamaño, bajo coste, y reducida capacidad de cómputo, almacenamiento, comunicación, y energía. Su arquitectura básica comprende cuatro módulos principales: el procesador, el transceptor (para comunicaciones), los sensores, y la alimentación (energética). El modelo básico de un nodo sensor es binario y se define en base a dos valores: el radio de medición R_{SENS} y el radio de comunicaciones R_{COMM}. Todo los que se encuentra a menor distancia que el radio de medición es medido (o detectado) por el nodo, así como cualquier nodo a menor distancia que el radio de comunicaciones puede recibir transmisiones desde el nodo. Existen dos tipos de arquitectura para redes de sensores, la plana, donde todos los nodos están al mismo nivel, y la jerárquica (o clusterizada). En nuestro trabajo se considera que la red es de tipo plano. Además, las redes tienen un nodo (o varios) especial llamado *High Energy Communications Node* (HECN), y que forma el punto de acceso a la red; todos los demás nodos de la red han de ser capaces de comunicarse, directa o indirectamente, con este nodo.

Las WSNs poseen características especiales que las distinguen de otros tipos de redes ad hoc. Deben trabajar de manera autónoma y por largos periodos de tiempo con escasa energía y sin mantenimiento. Albergan grandes cantidades de nodos sensores que deben configurarse automáticamente. Se despliegan en entornos hostiles, y deben responder a cambios en el entorno, o en la propia red.

Las WSNs ya han sido empleadas en multitud de tipos de aplicación. Los principales dominios en los que pueden clasificarse estas aplicaciones son los siguientes: aplicaciones militares, de vigilancia, ingeniería civil, aplicaciones biomédicas, servicios, industria, agricultura, medio ambiente, ayudas a zonas catastróficas, y exploración.

B.2.1 Problemas de optimización en redes de sensores

Como ya se anunció, el uso de WSNs implica la resolución de nuevos problemas de optimización. Ya existe un importante cuerpo de conocimiento en este sentido, por lo que realizamos una labor de compilación de los principales problemas encontrados:

- **Despliegue de nodos sensores (WSNL).** Este es uno de los problemas abordados en esta tesis doctoral. Consiste en decidir el número de nodos que se despliega así como las posiciones geográficas de cada uno de esos nodos, de manera que se consiga el grado de cobertura deseado, se maximice el tiempo de vida y se emplee el menor número de nodos posible.

- **Descubrimiento de localización (LD).** Otro de los problemas abordados en esta tesis doctoral. Dada una WSN ya desplegada y un conjunto de estimaciones de distancia entre pares de nodos, hay que encontrar las coordenadas geográficas correspondientes a los nodos de la red con la mayor precisión posible.

- **Planificación de tareas.** Es el tercer problema referenciado en esta tesis; si bien no se resuelve directamente, sí se resuelve el problema RND, que guarda relación con el mismo. Este problema consiste en realizar la planificación de tareas en la WSN de manera que se determina cuando cada nodo está activo y cuando en reposo. El objetivo es maximizar el tiempo de reposo de los distintos nodos para maximizar el tiempo de vida de la WSN, mientras se mantiene en todo momento las prestaciones requeridas (cobertura, conectividad).

- **Sincronización entre nodos.** Este problema consiste en generar una señal de reloj síncrona entre los distintos nodos de la red, con precisión suficiente para el correcto funcionamiento de los distintos procesos (mediciones, transmisión de información, etc.). Alternativamente, pueden idearse métodos de operación que sean robustos frente a errores de sincronía, o que no requieran sincronía en absoluto.

- **Control de topología.** Este problema consiste básicamente en garantizar la conectividad de la red, generalmente de manera distribuida y controlando la activación de los nodos así como la energía empleada para la transmisión de datos y por consiguiente el radio de comunicaciones. También pueden buscarse otras propiedades, como la planaridad del grafo.

- **Encaminamiento en WSNs.** En este problema se debe decidir la estrategia de routing seguida por los nodos (idealmente de manera totalmente distribuida), así como los valores de los parámetros empleados (si los hubiera). Los principales objetivos perseguidos en routing son la fiabilidad, eficiencia energética, y la latencia.

- **Data-fusion en WSNs.** Este problema consiste en integrar la información temporalmente, espacialmente, o ambas. Generalmente se persigue realizar un procesado de la información, o simplemente reducir la cantidad de información que se transmite (para ahorrar energía, evitar la congestión en la red, etc.).

- **Seguridad en WSNs.** La seguridad en WSNs incluye la protección de la información o la detección de intrusión (dentro de las comunicaciones de la red). También se estudian los efectos de ataques sobre algunos procesos de las redes (típicamente LD), así como algunas técnicas para la protección frente a dichos ataques.

En los distintos problemas descritos hay una serie de parámetros que surgen con cierta recurrencia y son objetivos que deben ser tenidos en cuenta. Los principales de ellos son: eficiencia energética (y tiempo de vida), probabilidad de detección y tasa de falsa alarma, latencia en la respuesta de la WSN, operación distribuida, uso eficiente de recursos limitados (de cómputo, comunicaciones y almacenamiento), robustez frente a condiciones adversas del entorno, y robustez frente a fallos de los nodos.

B.3 Metaheurísticas

Las metaheurísticas son estrategias de alto nivel que combinan distintos métodos para explorar un espacio de búsqueda. Suelen definirse a modo de plantillas que se deben rellenar empleando información específica

del problema sobre el cual han de aplicarse (representación de las soluciones, operadores, etc.), y son capaces de abordar problemas cuyos espacios de búsqueda son muy extensos. Las metaheurísticas pueden clasificarse dentro de dos categorías, según el número de soluciones que manejan de forma simultánea: las *basadas en trayectoria*, que tienen una única solución, y las *basadas en población*, que manejan un conjunto de soluciones, o población, de forma simultánea. Algunas metaheurísticas conocidas del primer tipo son el recocido simulado (SA), la búsqueda tabú (TS), GRASP, la búsqueda de vecindario variable (VNS), o la búsqueda local iterada (ILS). Algunos ejemplos conocidos del segundo tipo son los algoritmos evolutivos (EA), los algoritmos de estimación de distribuciones (EDA), la búsqueda dispersa (SS), la optimización por colonia de hormigas (ACO), y la optimización por cúmulos de partículas (PSO).

Hay dos características principales de los problemas seleccionados en esta tesis que deben ser tenidas en cuenta y que justifican el uso de metaheurísticas. La primera es que todos ellos implican una gran complejidad computacional, y por lo tanto requieren de muchos recursos para su resolución. La segunda es que, tanto en RND como en WSNL, existen distintos objetivos en conflicto, es decir que no pueden alcanzarse de manera simultánea, por lo que es necesaria una resolución de tipo multiobjetivo, en la que se persigue un cierto equilibrio entro los objetivos. Estas características nos impulsan a utilizar dos tipos avanzados de metaheurística: técnicas multiobjetivo basadas en la optimalidad de Pareto, y metaheurísticas paralelas para reducir los tiempos de cómputo. A continuación se presentan estas técnicas.

B.3.1 Técnicas multiobjetivo

En la optimización multiobjetivo se busca optimizar varios objetivos de manera simultanea, los cuales están en conflicto entre sí (de manera intuitiva, para mejorar uno se debe empeorar alguno de los otros). Por esto, y a diferencia de la optimización monoobjetivo, el óptimo no es una única solución, sino un conjunto de soluciones conocido como el óptimo de Pareto, el cual al ser representado en el espacio de objetivos da lugar al llamado *frente de Pareto*. Cada solución de este conjunto es óptima en el sentido de que no es posible mejorar ninguno de sus objetivos sin empeorar alguno de los demás.

El objetivo de la optimización multiobjetivo es pues la obtención del conjunto de soluciones Pareto-óptimas. No obstante, esto no siempre es factible; en ese caso el objetivo pasa a ser el obtener una aproximación suficientemente "buena" del conjunto, es decir, un conjunto de soluciones tal que se cumplen dos propiedades: cercanía al verdadero frente de Pareto, y diversidad de las soluciones a lo largo del frente.

B.3.2 Técnicas paralelas

A veces, los problemas resueltos por las metaheurísticas son tan complejos que los tiempos de computación resultan demasiado elevados. En estos casos, una posible opción es el uso de múltiples plataformas de cómputo de manera simultánea y cooperativa, de modo que el problema puede resolverse en un tiempo menor. Los algoritmos que siguen este tipo de estrategia se conocen como algoritmos paralelos.

En el caso de las metaheurísticas existen múltiples maneras de llevar a cabo la paralelización. Así, para las técnicas basadas en trayectoria, se tienen los modelos de ejecuciones múltiples, de movimientos paralelos, o de aceleración del movimiento, según si se ejecutan algoritmos completos en paralelo, si se realiza la exploración del vecindario de forma paralela, o si se fracciona el cálculo de la función de fitness y se realiza en paralelo, respectivamente. Para las metaheurísticas poblacionales, se tienen dos grandes modelos: las metaheurísticas distribuidas, en las cuales se divide la población en varias subpoblaciones de menor tamaño que se intercomunican, y las celulares en las cuales las soluciones se distribuyen siguiendo un modelo regular que establece vecindarios.

Estudio de la convergencia en EAs distribuidos

Según un estudio realizado por Luque y Alba [11], la convergencia de una población distribuida puede aproximarse, bajo ciertos supuestos, mediante la expresión:

$$P(t) = \sum_{i=1}^{i=d(T)} \frac{1/N}{1 + a \cdot e^{-b \cdot (t - per \cdot (i-1))}} + \frac{N - d(T)/N}{1 + a \cdot e^{-b \cdot (t - per \cdot d(T))}}, \tag{B.1}$$

donde $P(t)$ es la proporción de la población global ocupada por el óptimo, per es el periodo de migración, N es el número de subpoblaciones o islas, y $d(T)$ el diámetro de la topología. A partir de este resultado puede extraerse una expresión para el tiempo en que se alcanza la convergencia completa (el *takeover time*, $P(t) = 1$), como sigue:

$$t^* = per \cdot d(T) - \frac{1}{b} \cdot Ln \left(\frac{1}{a} \cdot \frac{\varepsilon}{N - d(T) - \varepsilon \cdot N} \right), \tag{B.2}$$

done t^* es el *takeover time*, medido con un nivel de precisión ε.

B.3.3 Algoritmos usados

Para resolver los distintos problemas abordados en esta tesis se emplea un juego de algoritmos con distintas características. Estos algoritmos son de tipo tanto monoobjetivo como multiobjetivo. Dentro del primero tipo, podemos mencionar la técnica basada en trayectoria SA, los algoritmos evolutivos GA y CHC, y la técnica basada en cúmulo de partículas PSO. Dentro del segundo tipo tenemos la técnica basada en trayectoria PAES, los evolutivos MOCHC, NSGA-II, y SPEA2, y el algoritmo celular MOCell. Adicionalmente, se emplea una versión paralela del GA para realizar el estudio de la efectividad de la técnica de migración autoadaptativa propuesta.

B.4 Diseño de la red de radio

Nuestro primero problema abordado es el problema de diseño de la red de radio (RND). Este problema consiste en seleccionar los emplazamientos para la colocación de estaciones base (o antenas), así como los posibles parámetros de las mismas, para ofrecer cobertura de radio a un determinado terreno, buscando colocar el mínimo número de estaciones base posible. Los emplazamientos deben ser escogidos entre una lista de emplazamientos disponibles. Los modelos más frecuentemente empleados para la estimación de la cobertura son los modelos de puntos de test, donde una serie de puntos especiales son definidos en los cuales se estima la recepción de la señal, y la rejilla o *grid*, en el cual se superpone una rejilla regular sobre el terreno y se evalúa la recepción de señal en cada punto de dicha rejilla.

Este problema puede relacionarse de forma sencilla con el problema de planificación de actividad y reposo en una WSN. En este ltimo problema debe decidirse los tiempos de actividad y de reposo de los nodos de manera que siempre se mantengan los niveles de cobertura, y los nodos estén en reposo el mayor tiempo posible. Este problema se puede transformar en elegir subconjuntos de nodos tales que cada subconjunto contiene el menor número de nodos posible, y mantiene la cobertura requerida; en este caso, elegir un subconjunto equivale a resolver RND.

B.4.1 Formulación

Planteamos el problema RND desde dos ópticas, la monoobjetivo y la multiobjetivo. En la primera, se busca maximizar al mismo tiempo cobertura y número de antenas; para ello, la formulación empleada es:

$$f(\vec{x}) = \frac{Coverage(\vec{x})^2}{|antennae|}, \tag{B.3}$$

donde la funcin f se debe maximizar. En la segunda, ambos objetivos se optimizan por separado:

$$f_1(\vec{x}) = 100 - Coverage(\vec{x}), \tag{B.4}$$

$$f_2(\vec{x}) = |antennae|, \tag{B.5}$$

donde ambas funciones se deben minimizar, y además se definen las siguientes restricciones para encaminar la búsqueda hacia zonas interesantes del espacio de búsqueda:

$$p_1(\vec{x}) = \begin{cases} f_1(\vec{x}) - K & (f_1(\vec{x}) > K) \\ 0 & (f_1(\vec{x}) \leq K) \end{cases}, \tag{B.6}$$

$$p_2(\vec{x}) = \begin{cases} f_2(\vec{x}) - N & (f_2(\vec{x}) > N) \\ 0 & (f_2(\vec{x}) \leq N) \end{cases}. \tag{B.7}$$

En nuestra definición del problema, la evaluación del terreno se realiza mediante el uso de una rejilla. Además, se emplean tres modelos distintos para la cobertura de las antenas: cobertura cuadrada y cobertura circular (sin parámetros), y cobertura sectorial o directiva (con parámetro dirección).

B.4.2 Resultados experimentales

Se abordó la resolución de 8 instancias de problema: 5 de tamaño reducido con antenas sin parámetros, 2 de tamaño medio con antenas directivas, y una de gran tamaño, basado en la ciudad de Málaga, usando antenas sin parámetros. En todos los casos excepto el último se planteó la resolución monoobjetivo así como la multiobjetivo. En cada prueba se realizan 30 ejecuciones independientes para realizar el análisis estadístico de los resultados.

En las 5 instancias de tamaño reducido las ejecuciones se realizaron hasta hallar el óptimo. Se encontró que el algoritmo CHC fue más eficiente que SA y que GA, y que su versión multiobjetivo, MOCHC, superó los resultados de NSGA-II, al encontrar el óptimo del problema en todos los casos necesitando un menor esfuerzo computacional que los demás algoritmos (visitando un menor número de soluciones del espacio de búsqueda). En las instancias con antenas directivas se reutilizó CHC/MOCHC con resultados satisfactorios, encontrándose que para algunos casos (los de mayor complejidad), la versión multiobjetivo es más efectiva que la monoobjetivo.

Para la instancia de Málaga las ejecuciones se realizaron hasta completar 5 millones de evaluaciones. Este trabajó se realizó dentro de un marco de colaboración con otros grupos, en un amplio estudio que incluyó hasta 14 distintas técnicas de optimización, quedando CHC dentro del primer cuartil (las más eficientes).

B.4.3 Técnica de migración automática

Nuestra propuesta de técnica de migración automática se basa en el estudio teórico de la convergencia en poblaciones distribuidas. La idea básica consiste en adaptar las migraciones de tal manera que la convergencia se produzca en el momento de terminación de la ejecución (parámetro fijo y predeterminado al comienzo de la prueba); de esta manera, al evitar tanto la convergencia prematura como la no convergencia, se espera alcanzar un equilibrio adecuado entre la exploración y la explotación de las soluciones.

Para esto, partiendo de la Ecuación B.2, podemos extraer:

$$t_{remaining} + per \cdot \left(\frac{P(t)}{1/N}\right) = per \cdot d(T) - \frac{1}{b} ln \left(\frac{1}{a} \frac{\epsilon}{N - d(T) - \epsilon N}\right), \tag{B.8}$$

a partir de la cual extraemos el periodo de migración per:

$$per = \frac{t_{remaining} - K}{d(T) - \left(\frac{P(T)}{1/N}\right)}, \tag{B.9}$$

donde se define:

$$K = \frac{1}{b} \cdot Ln\left(\frac{1}{a} \cdot \frac{\epsilon}{N - d(T) - \epsilon N}\right), \tag{B.10}$$

donde a es igual al tamaño de la población en una isla, $b = 0.4$, y ϵ es un factor de tolerancia que se configura como $\epsilon = 0.1$.

Evaluación experimental

Para la evaluación de la técnica de migración automática se compararon los resultados obtenidos por un GA distribuido que la incorpora frente a 10 GAs distribuidos con configuraciones fijas de migración, y 2 GAs secuenciales. El modelo escogido de población distribuida fue de 8 islas de 50 individuos (población global de 400 individuos), formando una topología de anillo unidireccional. Se utilizaron dos modalidades según los operadores de selección: la primera, llamada Elitista, combina una selección por ruleta y un reemplazo elitista; la segunda, llamada Normal, combina una selección aleatoria y un reemplazo por torneo. Para este experimento, las ejecuciones se realizaron hasta realizar 5 millones de evaluaciones.

Los resultados demostraron que, si bien nuestra técnica propuesta no obtuvo los mejores resultados entre todos los algoritmos distribuidos, sí que obtuvo resultados cuando menos comparables; la técnica propuesta funcionó mejor con la configuración de selección denominada Elitista. Comparada a los algoritmos secuenciales, los resultados fueron algo inferiores, pero se compensa por un menor tiempo para la obtención de resultados de calidad similar.

Finalmente, se estimó que el ahorro debido a la configuración automática de los parámetros de migración puede suponer entre un 75% y un 89% del tiempo completo de realización de la prueba.

B.5 Despliegue de nodos sensores

El segundo problema abordado es el despliegue de nodos para formar una red de sensores (WSNL). En este problema, hay que determinar la cantidad de nodos que se va a desplegar así como las coordenadas en las que se va a colocar cada uno de ellos. Los objetivos perseguidos son obtener la mayor cobertura posible (o bien un determinado grado de cobertura predeterminado), formar una red conexa con el HECN, emplear el menor número de nodos posible, y obtener el mayor tiempo de vida posible (se asume que el gasto energético se debe a las comunicaciones de información).

Para este problema se requiere el uso de modelos de cobertura y comunicaciones, así como una definición del tiempo de vida. Los modelos más conocidos para la cobertura a nivel de un nodo son la cobertura binaria, la cuasi-binaria, y la probabilística; para la cobertura a nivel de red se conocen la cobertura de puntos, cobertura de área, cobertura de perímetro, cobertura de camino, o cobertura diferenciada (con grados de detección), entre otras; en nuestro caso empleamos cobertura binaria para nodo, de área para la red, y empleamos un grid para el cómputo. Para las comunicaciones se tienen los mismos modelos a nivel de nodo, que luego han de combinarse con la jerarquía de red y el protocolo de routing empleado. Nosotros asumimos un modelo binario, red plana, y proponemos un routing de equilibrio energético local, donde cada nodo distribuye la información entre los nodos más cercanos al HECN de manera proporcional al inverso de la potencia necesaria para la transmisión.

Asumimos el criterio de *time to first failure* (TTFF) para la estimación del tiempo de vida, es decir que el tiempo de vida abarca hasta el momento en que el primer nodo se queda sin energía. El consumo de energía se debe exclusivamente a las transmisiones de información, y el modelo que empleamos para la potencia de transmisión es cuadrático (la potencia de transmisión es proporcional al cuadrado de la distancia del enlace: $P \propto d^2$).

B.5.1 Formulación

En nuestro acercamiento al problema adoptamos una formulación multiobjetivo. En ella, tanto el número de nodos sensores como el tiempo de vida son objetivos que hay que optimizar, mientras que la cobertura se impone como restricción (exigimos 100% de cobertura). Por lo tanto, la formulación es la siguiente, donde los objetivos son:

$$f_1(\vec{x}) = Cost(\vec{x}), \tag{B.11}$$
$$f_2(\vec{x}) = Energy(\vec{x}), \tag{B.12}$$

sujeto a la restricción impuesta por la función de penalización P:

$$P(\vec{x}) = 100 - C(\vec{x}). \tag{B.13}$$

Asumimos que los nodos empleados tienen valores $R_{SENS} = R_{COMM} = 30m$.

B.5.2 El operador de mejora PACO

Proponemos un operador de mejora local para las soluciones candidatas, el *Proximity Avoidance Coverage-preserving Operator* (PACO). El principio básico de funcionamiento de este operador consiste en buscar parejas de nodos cercanos, e intentar reemplazar la pareja de nodos por un único nodo tal que se mantengan la cobertura y la conectividad de la red tras el cambio. De esta manera, el número de nodos se habrá reducido (mejora en un objetivo), y el tráfico de la nueva red será menor –al haber un nodo menos–, por lo que podría haber menor consumo de energía (posible mejora en el segundo objetivo).

El funcionamiento es el siguiente: una vez localizada la pareja de nodos próximos, se identifica el área que es cubierta exclusivamente por eso nodos, así como los nodos de los que *cuelgan*, y aquellos que a su vez cuelgan de ellos. Se establecen las áreas equivalentes para cobertura y conectividad, tales que un único nodo colocado en esas areas asegura la cobertura del área (caso del area equivalente de cobertura), y la conexión con *todos* los nodos hijos de la pareja, y con *al menos uno* de los padres (nótese que cualquiera de estas reas podra no existir). Si ambas areas tienen zona común, se sustituye la pareja por un único nodo que se coloca en esa zona común.

El operador se integra dentro de los algoritmos a continuación de la etapa de evaluación de las nuevas soluciones (etapa que todas las técnicas tienen). Además, cada vez que el operador PACO sustituye una pareja de nodos por un nodo lleva a cabo un reevaluación de la solución, de manera que únicamente da por buenos los cambios que no empeoran ningún objetivo (criterio elitista). Además, estas evaluaciones son contabilizadas dentro de la ejecución de los algoritmos, de manera que el esfuerzo computacional de una ejecución no se modifica debido al uso de PACO, por lo que las comparaciones entre técnicas con PACO y técnicas sin PACO son válidas.

B.5.3 Resultados experimentales

Para los experimentos con este problema se escogieron 4 algoritmos multiobjetivo del estado del arte: los evolutivos NSGA-II y SPEA2, el basado en trayectoria PAES, y el celular MOCell. Cada uno se configuró con todas las combinaciones posibles de operadores aleatorios y geográficos, y tanto usando PACO como no usándolo. Se resolvieron instancias de 3 tamaños distintos (con máximos de 500, 1000, y 2000 nodos respectivamente), y cada escenario fue resuelto 30 veces de manera independiente.

En la instancia de problema básica (hasta 500 nodos), la efectividad del operador de mejora PACO quedó demostrada al obtener mejores valores de hipervolumen en el 84.85% del total de configuraciones probadas. Sin embargo, cuando la comparación se restringe únicamente al 50% de configuraciones con mejores resultados, el uso de PACO supone una mejora en el 98.48% de los escenarios. Por lo tanto, nuestro operador tiene un comportamiento muy robusto, que se ve acrecentado cuando la configuración

algorítmica de base es una configuración de buenos resultados. Respecto a los operadores, observamos una clara ventaja de los de tipo geográfico frente a los aleatorios, y entre los algoritmos MOCell y NSGA-II fueron los que obtuvieron los mejores resultados (algo mejores por parte de MOCell, pero mayor robustez frente a distintas configuraciones en NSGA-II).

Al comprobar la efectividad del operador para instancias de problema de mayor complejidad (hasta 1000 o 2000 nodos), es decir la escalabilidad del mismo, observamos que a medida que la instancia de problema aumenta en complejidad, los resultados de PACO tienden a mejorar. Esto se aprecia en el hecho de que, mientras que las configuraciones sin PACO se ven seriamente degradadas al aumentar la complejidad del problema, aquellas que incorporan PACO se mantienen mucho mejor. Es notable el caso particular de PAES, que pasa a ser el mejor algoritmo en la instancia de mayor tamaño.

B.6 Descubrimiento de localización

El tercer problema abordado es el de descubrimiento de la localización (LD). En este problema, los nodos, que ya han sido desplegados, deben averiguar las coordenadas de sus posiciones geográficas, de manera que puedan dotar de sentido espacial a las mediciones que realizan. Para ello, se dispone de una serie de mediciones de distancia entre pares de nodos, así como de un pequeño subconjunto de nodos equipados con algún sistema de autolocalización que conocen sus coordenadas y sirven como referencia para el resto de nodos de la red, llamados balizas o *beacon nodes*.

Los métodos existentes para la medición de distancias involucran el uso de señales radio y/o señales acústicas. Los principales que se recogen en la literatura son: el nivel de intensidad de señal recibida, RSSI (para la señal de radio), el tiempo de llegada, ToA (usando señal acústica), y la diferencia de tiempos de llegada, TDoA (combinando ambas señales).

B.6.1 Formulación

Nuestro problema tiene como objetivo encontrar las coordenadas de los nodos con el menor error posible, esto es, queremos minimizar la función:

$$fitness(\vec{x}) = \frac{\sum_{i=1}^{|\vec{x}|} |x_i - n_i|}{|\vec{x}|}, \tag{B.14}$$

donde x_i es la posición determinada para el nodo i, cuya auténtica posición es n_i.

Sin embargo, no es posible emplear esta función para guiar la búsqueda ya que requiere conocimiento sobre las auténticas posiciones, lo cual no es posible en un escenario real. Por lo tanto, consideramos dos funciones de guiado para nuestro acercamiento a este problema:

- Funciones de tipo norma de error. Estas funciones realizan una valoración del error cometido por una solución determinada, en base a la diferencia entre las distancias medidas entre parejas de nodos, y las distancias resultantes de las posiciones determinadas. Concretamente, la función L_1 se define como:

  $$L_1' = |\omega(\delta_1) \cdot \epsilon_1| + |\omega(\delta_2) \cdot \epsilon_2| + |\omega(\delta_3) \cdot \epsilon_3| + ... + |\omega(\delta_L) \cdot \epsilon_L|, \tag{B.15}$$

  donde ϵ_i es la diferencia entre ambos valores de distancia para el enlace i. La función está sujeta a minimización. Además asociamos un valor de peso ω a cada componente de error que pondera la calidad esperada de la medición; estos pesos se obtienen a partir de información específica del problema, o, para ser más exactos, conocimiento específico de la técnica de medición.

- Funciones de probabilidad. Estas funciones asignan a cada solución una probabilidad de que dicha solución sea correcta, en base a la relación entre las distancias medidas y las distancias obtenidas

como fruto de la asignación de coordenadas a los nodos. Para esto es necesario tener conocimiento del problema suficiente para generar una función de densidad de probabilidad para estas dos magnitudes.

$$L = P(d_1, \delta_1) * P(d_2, \delta_2) * P(d_3, \delta_3) * ... * P(d_L, \delta_L). \quad \text{(B.16)}$$

B.6.2 Resolución en dos fases

Llevamos a cabo un estudio de la consistencia de los dos tipos de función de guía, es decir, de la correlación entre la solución favorecida por la función de guía y la solución que tiene menor error de localización para distintos pares de soluciones, en diferentes condiciones. Como resultado del estudio observamos que cuando las dos soluciones tienen un elevado componente de error (para soluciones dichas "aleatorias"), la función L_1 alcanza una consistencia del 62.1% frente al 55.8% de la función de probabilidad, mientras que para soluciones con baja componente de error esos valores son de 51.9% (cuasi aleatoriedad) frente a 71.7% respectivamente.

Como consecuencia, proponemos una resolución en dos fases. Durante la primera fase, en la cual la solución tiene un elevado componente de error, se emplea como guía la función L_1. Un vez concluida la primera fase, y obtenida (en principio) una solución con baja componente de error, comienza la segunda fase, que utiliza la función de probabilidad como guía. De esta manera se pretende emplear en cada etapa de la búsqueda aquella función de guiado que ofrece las mejores prestaciones.

B.6.3 Resultados experimentales

Para poner a prueba nuestra propuesta de resolución en dos fases disponemos de 33 conjuntos de datos *reales* obtenidos de otras tantas mediciones llevadas a cabo sobre WSNs desplegada en el fuerte Leonard Wood (EE.UU.), cuyos números de nodos varían entre 79 y 93. Seleccionamos 10 conjuntos de datos para que sean las instancias de problema, mientras que el resto sirve como base para establecer los modelos de pesos de los enlaces, y de función de probabilidad. Para cada instancia generamos 10 configuraciones distintas de balizas, y cada escenario se resuelve 100 veces de manera independiente para producir los resultados. Como algoritmos seleccionamos SA, GA y PSO, y por defecto fijamos el número de balizas al 10%. Las ejecuciones se detienen al evaluar 5 millones de soluciones.

Las primeras pruebas se realizan empleando la función L_1 como guía. Los primeros resultados demuestran la efectividad de los pesos combinados con la función L_1, ya que producen una notable mejora en los errores de localización en la mitad de las instancias, mientras que para la otra mitad los errores son similares o ligeramente mejores. Al estudiar la influencia del número de balizas, observamos que si aumentamos la proporción de balizas hasta el 20% el error de localización se reduce enormemente, y se vuelve muy estable; si se aumenta desde el 20 hasta el 30%, por contra, no se aprecian grandes variaciones. Por lo tanto, deducimos que un buen valor de compromiso es 20% de balizas; no obstante, y con el ánimo de afrontar un desafío mayor, enfrentamos nuestras técnicas a instancias de problema con sólo un 10% de balizas. Al comparar los algoritmos se aprecia una clara ventaja de SA frente a los otros dos algoritmos, que obtienen resultados similares.

Finalmente comparamos nuestra técnica propuesta, la de dos fases, frente a la resolución utilizando únicamente L_1, y utilizando únicamente la función de probabilidad. En nuestra configuración, la primera fase acaba tras evaluar 4 millones de soluciones, y la segunda fase realiza un millón de evaluaciones más (de manera que el número total de soluciones evaluadas se mantiene en 5 millones). Los resultados son contundentes: en el 94% de los casos de test la resolución en dos fases supera a ambas resoluciones que usan una única función de guiado. Por lo tanto, concluimos que nuestra propuesta resulta efectiva.

B.7 Conclusiones

En esta tesis doctoral hemos abordado la resolución de dos de los principales problemas en redes de sensores, el despliegue de nodos (WSNL) y el descubrimiento de localización (LD), y de un tercer problema, el diseño de una red de radio (RND), directamente relacionado con otro de los problemas de redes de sensores, el de planificación. La resolución de estos problemas se ha hecho empleando diversas técnicas metaheurísticas, entre las cuales hay técnicas basadas en trayectoria, poblacionales, basadas en partículas, y celulares. Hemos empleado formulaciones tanto monoobjetivo (RND y LD) como multiobjetivo (RND y WSNL), y un algoritmo paralelo (RND).

El problema de diseño de la red de radio (RND) consiste en escoger las localizaciones para la colocación de antenas –de entre un conjunto de localizaciones disponibles–, así como los parámetros de configuración de las mismas (si los hay), para obtener la mayor cobertura empleando el menor número de antenas. Cuando se emplean antenas sin parámetros (codificación binaria), los mejores resultados se obtienen por CHC en el planteamiento monoobjetivo y por MOCHC en el multiobjetivo. Cuando se usan antenas directivas (con parámetro dirección), CHC y MOCHC siguen produciendo buenas soluciones. En un amplio estudio realizado sobre una instancia de gran dimensión basada en la ciudad de Málaga, CHC quedó dentro del primer cuartil (entre un total de 14 técnicas). Proponemos una técnica automática para controlar las migraciones en un GA distribuido, que obtiene resultados similares a los mejores resultados encontrados usando migraciones periódicas y sólo ligeramente inferiores a los del GA secuencial equivalente, pero reduciendo el tiempo completo de la prueba entre un 75% y un 89%.

El problema de despliegue de los nodos sensores (WSNL) consiste en determinar el número de nodos y sus posiciones para obtener la mayor cobertura y tiempo de vida empleando el menor número de nodos posible. Proponemos un operador de mejora local, PACO, que busca resolver pequeñas ineficiencias debidas a parejas de nodos cercanos. Tras probarla con 4 algoritmos multiobjetivos y distintas configuraciones algorítmicas, nuestra propuesta resulta efectiva en un 84.85% del total de configuraciones, y en un 98.48% de las configuraciones con mejor rendimiento. Más aún, cuando el tamaño de la instancia de problema aumenta, las ventajas de utilizar PACO se vuelven mayores.

El problema de descubrimiento de la localización (LD) consiste en averiguar las posiciones de los nodos basándose en una serie de distancias medidas entre parejas de nodos, y un subconjunto de nodos cuyas posiciones son conocidas. Tras estudiar las dos principales funciones de guiado existentes, la norma del error y la probabilidad, se propone un sistema en dos fases para la resolución del problema. Empleando un conjunto de 10 instancias creadas con datos –mediciones– reales, nuestra técnica propuesta proporciona mejores resultados en el 94% de los escenarios de test.

La presente tesis doctoral ha tenido una notable repercusión mediante las publicaciones realizadas en foros de divulgación, con especial atención a las revistas de impacto, para los diferentes temas abordados. La validez de CHC en la resolución de RND se presenta en [3][1], la efectividad del uso de metaheurísticas para resolver WSNL queda demostrada en [4], la efectividad de PACO en [1], mientras que los beneficios del uso combinado de distintas técnicas de guiado en el problema LD se expone en [2]. Por último, cabe resaltar que el trabajo realizado en esta tesis no se reduce a la resolución de problemas puramente académicos, y tiene una fuerte componente de aplicabilidad. Así, el trabajo realizado en RND puede emplearse para la selección de estaciones base de una red de telefonía celular, los resultados de LD pueden emplearse para la geolocalización de nodos o terminales móviles, y los resultados de WSNL pueden servir para el diseño de estrategias de despliegue de nodos sensores. Futuras líneas avanzarán en el modelado realista del problema y el desarrollo de técnicas distribuidas y ligeras especialmente adaptadas para su ejecución en la plataforma que representan los nodos sensores.

[1] Las citas están referidas a las publicaciones propias, que pueden verse en el capítulo dedicado.

Index

Bibliography

[1] K. I. Aardal, S. P. M. van Hoesel, A. M. C. A. Koster, C. Mannino, and A. Sassano. Models and solution techniques for frequency assignment problems. *Annals of Operations Research*, 153(1):79 – 129, 2007.

[2] J. Ai and A. A. Abouzeid. Coverage by directional sensors in randomly deployed wireless sensor networks. *J. Comb. Optim.*, 11(1):21–41, 2006.

[3] N. Aitsaadi, N. Achirt, K. Boussetta, and G. Pujolle. A tabu search approach for differentiated sensor network deployment. In *Proceedings of the 5th IEEE Consumer Communications and Networking Conference, 2008. CCNC 2008.*, pages 163 –167. IEEE Computer Society Press, jan. 2008.

[4] I. Akyildiz, W. Su, Y. Sankasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 2002.

[5] E. Alba. *Anlisis y Diseo de Algoritmos Genticos Paralelos Distribuidos*. PhD thesis, University of Mlaga, 1999.

[6] E. Alba, editor. *Parallel Metaheuristics: A New Class of Algorithms*. Wiley, 2005.

[7] E. Alba. Evolutionary algorithms for optimal placement of antennae in radio network design. *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, pages 168–, 26-30 April 2004.

[8] E. Alba and F. Chicano. On the behavior of parallel genetic algorithms for optimal placement of antennae in telecommunications. *International Journal of Foundations of Computer Science*, 16(2):343–359, April 2005.

[9] E. Alba, C. Cotta, F. Chicano, and A. J. Nebro. Parallel evolutionary algorithms in telecommunications: Two case studies. In *Proceedings of the Congreso Argentino de Ciencias de la Computación (CACIC02)*, Buenos Aires, Argentina, 2002.

[10] E. Alba, F. Luna, and A. J. Nebro. Advances in parallel heterogeneous genetic algorithms for continuous optimization. *International Journal of Applied Mathematics and Computer Science*, 14(3):101 – 117, 2004.

[11] E. Alba and G. Luque. Theoretical models of selection pressure for dEAs: Topology influence. In *IEEE Congress on Evolutionary Computation CEC-05*, pages 214–222, Edinburgh, UK, September 2005. IEEE Press.

[12] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443 – 462, 2002.

[13] E. Amaldi, A. Capone, and F. Malucelli. Planning umts base station location: optimization models with power control and algorithms. *Wireless Communications, IEEE Transactions on*, 2(5):939–952, Sept. 2003.

[14] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: a wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605 – 634, 2004. Military Communications Systems and Technologies.

[15] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996.

[16] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, New York, 1996.

[17] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T. H. Lai. Deploying wireless sensors to achieve both coverage and connectivity. In *MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 131–142, New York, NY, USA, 2006. ACM.

[18] A. Bharathidasan, V. An, and S. Ponduru. Sensor networks: An overview. Technical report, Department of Computer Science, University of California, Davis, 2002.

[19] E. S. Biagioni and K. W. Bridges. The application of remote sensor technology to assist the recovery of rare and endangered species. *International Journal of High Performance Computing Applications*, 16:2002, 2002.

[20] E. S. Biagioni and G. Sasaki. Wireless sensor placement for reliable and efficient data collection. In *Proceedings of the Hawaii International Conference on Systems Sciences, HICSS 03*, volume 5, page 127b, Los Alamitos, CA, USA, Jan 2003. IEEE Computer Society.

[21] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.

[22] T. Bokareva. Mini hardware survey. "http://www.cse.unsw.edu.au/s̃ensar/hardware/hardware_survey.html".

[23] A. Bonivento, L. P. Carloni, and A. L. Sangiovanni-Vincentelli. Platform-based design of wireless sensor networks for industrial applications. In *DATE*, pages 1103–1107, 2006.

[24] B. Bougard, F. Catthoor, D. C. Daly, A. Chandrakasan, and W. Dehaene. Energy efficiency of the ieee 802.15.4 standard in dense wireless microsensor networks: modeling and improvement perspectives. In *Proceedings of Design, Automation and Test in Europe, 2005.*, pages 196 – 201 Vol. 1, march 2005.

[25] P. Brass. Bounds on coverage and target detection capabilities for models of networks of mobile sensors. *ACM Trans. Sen. Netw.*, 3(2):9, 2007.

[26] J. Bruck, J. Gao, and A. A. Jiang. Localization and routing in sensor networks by local angle information. *ACM Trans. Sen. Netw.*, 5(1):1–31, 2009.

[27] J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing: Sensor networks in agricultural production. *IEEE Pervasive Computing*, 3:38–45, 2004.

[28] S. Cahon, N. Melab, and E.-G. Talbi. Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *J. Heuristics*, 10(3):357–380, 2004.

[29] S. Cahon, N. Melab, E.-G. Talbi, and M. Schoenauer. ParaDisEO-based design of parallel and distributed evolutionary algorithms. In *Artificial Evolution*, pages 216–228, 2003.

[30] P. Calégari, F. Guidec, and P. Kuonen. A Parallel Genetic Approach to Transceiver Placement Optimisation. In C.-A. Hritier and B. Chopard, editors, *Proceedings of the SIPAR Workshop'96: Parallel and Distributed Systems*, pages 21–24, October 1996.

[31] P. Calégari, F. Guidec, P. Kuonen, and D. Kobler. Parallel island-based genetic algorithm for radio network design. *Journal of Parallel and Distributed Computing*, 47(1):86–90, 1997.

[32] P. Calégari, F. Guidec, P. Kuonen, and F. Nielsen. Combinatorial optimization algorithms for radio network planning. *Theoretical Computer Science*, 263(1-2):235–265, 2001.

[33] P. Calégari, F. Guidec, P. Kuonen, and D. Wagner. Genetic approach to radio network optimization for mobile systems. In *Proceedings of the 47th Vehicular Technology Conference*, volume 2, pages 755–759, Phoenix, AZ, USA, May 1997. IEEE Computer Society.

[34] M. Cardei and J. Wu. Energy-efficient coverage problems in wireless ad-hoc sensor networks. *Computer Communications*, 29(4):413–420, 2006.

[35] V. Cevher and L. M.Kaplan. Acoustic sensor network design for position estimation. *ACM Trans. Sen. Netw.*, 5(3):1–28, 2009.

[36] A. Chakrabarti, A. Sabharwal, and B. Aazhang. Communication power optimization in a sensor network with a path-constrained mobile observer. *ACM Trans. Sen. Netw.*, 2(3):297–324, 2006.

[37] K. Chebrolu, B. Raman, N. Mishra, P. K. Valiveti, and R. Kumar. Brimon: a sensor network system for railway bridge monitoring. In *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*, pages 2–14, New York, NY, USA, 2008. ACM.

[38] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *ACM Wireless Networks*, 8(5), September 2002.

[39] J. Chen, Y. Guan, and U. Pooch. Customizing a geographical routing protocol for wireless sensor networks. In *ITCC '05: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II*, pages 586–591, Washington, DC, USA, 2005. IEEE Computer Society.

[40] Y. Chen, C.-N. Chuah, and Q. Zhao. Sensor placement for maximizing lifetime per unit cost in wireless sensor networks. *Military Communications Conference, 2005. MILCOM 2005. IEEE*, pages 1097–1102 Vol. 2, 17-20 Oct. 2005.

[41] Y. Chen, K. Kleisouris, X. Li, W. Trappe, and R. P. Martin. A security and robustness performance analysis of localization algorithms to signal strength attacks. *ACM Trans. Sen. Netw.*, 5(1):1–37, 2009.

[42] M. X. Cheng, L. Ruan, and W. Wu. Achieving minimum coverage breach under bandwidth constraints in wireless sensor networks. In *Proceedings IEEE of INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 4, pages 2638 – 2645, march 2005.

[43] M. X. Cheng, L. Ruan, and W. Wu. Coverage breach problems in bandwidth-constrained sensor networks. *ACM Trans. Sen. Netw.*, 3(2):12, 2007.

[44] J. F. Chicano. *Metaheurísticas e Ingeniería del Software*. PhD thesis, University of Mlaga, 2007.

[45] L. Chitnis, A. Dobra, and S. Ranka. Aggregation methods for large-scale sensor networks. *ACM Trans. Sen. Netw.*, 4(2):1–36, 2008.

[46] J.-C. Choi and C.-W. Lee. Energy modeling for the cluster-based sensor networks. In *The Sixth IEEE International Conference on Computer and Information Technology, 2006. CIT '06.*, pages 218 –218, sept. 2006.

[47] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, second edition, September 2007. ISBN 978-0-387-33254-3.

[48] J. L. Cohon and D. H. Marks. A review and evaluation of multiobjective programming techniques. *Water Resources Research*, 11(2):208 – 220, 1975.

[49] J. A. Costa, N. Patwari, and I. Alfred O. Hero. Distributed weighted-multidimensional scaling for node localization in sensor networks. *ACM Trans. Sen. Netw.*, 2(1):39–64, 2006.

[50] T. G. Crainic and M. Toulouse. Parallel strategies for metaheuristics. In F. W. Glover and G. A. Kochenberger, editors, *Handbook of Metaheuristics*, Norwell, MA, USA, 2003. Kluwer Academic Publishers.

[51] J. Créput, A. Koukam, T. Lissajoux, and A. Caminada. Automatic mesh generation for mobile network dimensioning using evolutionary approach. *IEEE Trans. Evolutionary Computation*, 9(1):18–30, 2005.

[52] B. Cărbunar, A. Grama, J. Vitek, and O. Cărbunar. Redundancy and coverage detection in sensor networks. *ACM Trans. Sen. Netw.*, 2(1):94–128, 2006.

[53] D. Culler, D. Estrin, and M. Srivastava. Overview of sensor networks. *IEEE Computer*, 37(8):41–49, 2004.

[54] V.-D. Cung, S. L. Martins, C. C. Ribeiro, and C. Roucairol. Strategies for the Parallel Implementation of Metaheuristics. In C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 263–308, Norwell, MA, USA, 2003. Kluwer Academic Publishers.

[55] K. Deb. *Optimization for Engineering Design*. Prentice-Hall, New Delhi, 1995.

[56] K. Deb. An efficient constraint handling mechanism method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2/4):311 – 338, 2000.

[57] K. Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001.

[58] K. Deb and R. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148, 1995.

[59] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, Apr 2002.

[60] J. Demšar. Statistical comparison of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1 – 30, 2006.

[61] S. Dhillon and K. Chakrabarty. Sensor placement for effective coverage and surveillance in distributed sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, pages 1609–1614, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[62] I. Dietrich and F. Dressler. On the lifetime of wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(1):1–39, 2009.

[63] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, DEI, Politecnico di Milano, Italy, 1992. (in italian).

[64] M. Dorigo and T. Stützle. *Handbook of Metaheuristics*, volume 57 of *International Series In Operations Research and Management Science*, chapter The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances, pages 251 – 285. Kluwer Academic Publisher, 2003.

[65] B. Dorronsoro. *Diseño e implementación de algoritmos genéticos celulares para problemas complejos*. PhD thesis, University of Málaga, 2007.

[66] M. Drinic, D. Kirovski, and M. Potkonjak. Model-based compression in wireless ad hoc networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 231–242, New York, NY, USA, 2003. ACM.

[67] P. Dubois, C. Botteron, V. Mitev, C. Menon, P.-A. Farine, P. Dainesi, A. Ionescu, and H. Shea. Ad-Hoc Wireless Sensor Networks For Exploration Of Solar-System Bodies. *Acta Astronautica*, Volume 64(Issues 5-6):470 – 478, 2009.

[68] M. Ehrgott. *Multicriteria Optimization*. Springer, second edition, 2005.

[69] L. J. Eshelman. The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. In *Foundations of Genetic Algorithms*, pages 265–283. Morgan Kaufmann, 1991.

[70] M. Esseghir, N. Bouabdallah, and G. Pujolle. Sensor placement for maximizing wireless sensor network lifetime. In *Vehicular Technology Conference, 2005. VTC-2005-Fall. 2005 IEEE 62nd*, volume 4, pages 2347 – 2351, sept. 2005.

[71] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: scalable coordination in sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 263–270, New York, NY, USA, 1999. ACM.

[72] L. Fang and W. Du. A beacon-less location discovery scheme for wireless sensor networks. In *In Proceedings of IEEE INFOCOM*, pages 13–17, 2005.

[73] J. Feng, L. Girod, and M. Potkonjak. Location discovery using data-driven statistical error modeling. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–14, April 2006.

[74] T. Feo and M. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109 – 133, 1999.

[75] K. P. Ferentinos and T. A. Tsiligiridis. A memetic algorithm for optimal dynamic design of wireless sensor networks. *Comput. Commun.*, 33(2):250–258, 2010.

[76] P. Floréen, P. Kaski, T. Musto, and J. Suomela. Local approximation algorithms for scheduling problems in sensor networks. In *ALGOSENSORS*, pages 99–113, 2007.

[77] P. Floréen, P. Kaski, and J. Suomela. A distributed approximation scheme for sleep sceduling in sensor networks. In *SECON*, pages 152–161, 2007.

[78] C. Fonseca and P. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Genetic Algorithms: Proceedings of the Fifth International Conference*, pages 416–423. Morgan Kaufmann, 1993.

[79] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proc. of the Fifth Int. Conference on Genetic Algorithms*, pages 416 – 423, 1993.

[80] D. Ganesan, R. Cristescu, and B. Beferull-Lozano. Power-efficient sensor placement and transmission structure for data gathering under distortion constraints. *ACM Trans. Sen. Netw.*, 2(2):155–181, 2006.

[81] L. Girod. Development and characterization of an acoustic rangefinder, 2000.

[82] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8:156 – 166, 1977.

[83] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13:533–549, 1986.

[84] F. Glover. A template for Scatter Search and Path Relinking. In J.-K. H. et al., editor, *Artificial Evolution*, number 1363 in LNCS, pages 13–54. Springer, 1998.

[85] F. W. Glover and G. A. Kochenberger. *Handbook of Metaheuristics*. Kluwer, 2003.

[86] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[87] R. L. Graham. Bounds on multiprocessor timing anomalies. *SIAM Journal of Applied Mathematics*, 17:416 – 429, 1969.

[88] T. C. E. Group. The sensor network museum. http://www.snm.ethz.ch/Main/HomePage.

[89] G. Gupta and M. Younis. Fault-tolerant clustering of wireless sensor networks. In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, volume 3, pages 1579 –1584 vol.3, march 2003.

[90] H. Gupta, V. Navda, S. Das, and V. Chowdhary. Efficient gathering of correlated data in sensor networks. *ACM Trans. Sen. Netw.*, 4(1):1–31, 2008.

[91] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh. Vigilnet: An integrated sensor network system for energy-efcient surveillance. *ACM Trans. Sen. Netw.*, 2(1):1–38, 2006.

[92] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 2000.*, page 10 pp. vol.2, jan. 2000.

[93] Y. Hochberg and A. C. Tamhane. *Multiple Comparison Procedures*. Wiley, 1987.

[94] T. T. Hsieh. Using sensor networks for highway and traffic applications. In *IEEE Potentials*, volume 23, pages 13–16, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[95] C.-F. Huang, L.-C. Lo, and Y.-C. Tseng. Decentralized energy-conserving and coverage-preserving protocols for wireless sensor networks. *ACM Trans. Sen. Netw.*, 2(2):182–187, 2006.

[96] C.-F. Huang, Y.-C. Tseng, and H.-L. Wu. Distributed protocols for ensuring both coverage and connectivity of a wireless sensor network. *ACM Trans. Sen. Netw.*, 3(1):5, 2007.

[97] N. Jaggi, K. Kar, and A. Krishnamurthy. Near-optimal activation policies in rechargeable sensor networks under spatial correlations. *ACM Trans. Sen. Netw.*, 4(3):1–36, 2008.

[98] E. Jain and Q. Liang. Sensor placement and lifetime of wireless sensor networks: theory and performance analysis. In *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, volume 1, pages 173–177, nov.-2 dec. 2005.

[99] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[100] I. Johnstone, J. Nicholson, B. Shehzad, and J. Slipp. Experiences from a wireless sensor network deployment in a petroleum environment. In *IWCMC '07: Proceedings of the 2007 international conference on Wireless communications and mobile computing*, pages 382–387, New York, NY, USA, 2007. ACM.

[101] D. Jourdan and O. de Weck. Layout optimization for a wireless sensor network using a multi-objective genetic algorithm. In *Proceedings of the IEEE Semiannual Vehicular Technology Conference*, volume 5, pages 2466–2470, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[102] D. B. Jourdan and N. Roy. Optimal sensor placement for agent localization. *ACM Trans. Sen. Netw.*, 4(3):1–40, 2008.

[103] D. Jung, T. Teixeira, and A. Savvides. Sensor node lifetime analysis: Models and tools. *ACM Trans. Sen. Netw.*, 5(1):1–33, 2009.

[104] Y. S. K. H.-M. Jung. Efficient radio network optimization. *Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual*, 3:1546–1549 vol.3, 22-25 April 2003.

[105] C.-W. Kang and J.-H. Chen. Multi-objective evolutionary optimization of 3d differentiated sensor network deployment. In *GECCO '09: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference*, pages 2059–2064, New York, NY, USA, 2009. ACM.

[106] K. Kar and S. Banerjee. Node placement for connected coverage in sensor networks. In *Proceedings of WiOpt*, 2003.

[107] S. Kar and J. M. F. Moura. Consensus based detection in sensor networks: Topology optimization under practical constraints. In *Proceedings of the 1st Internation Workshop on Information Theory in Sensor Networks (WITS)*, 2007.

[108] K. Karenos, V. Kalogeraki, and S. V. Krishnamurthy. Cluster-based congestion control for sensor networks. *ACM Trans. Sen. Netw.*, 4(1):1–39, 2008.

[109] L. Karim, N. Nasser, and T. Sheltami. A fault tolerant dynamic clustering protocol of wireless sensor networks. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1 –6, 30 2009-dec. 4 2009.

[110] H. Karl and A. Willig. A short survey of wireless sensor networks. Technical Report TKN-03-018, Technical University Berlin, 2003.

[111] A. H. Karp and H. P. Flatt. Measuring parallel processor performance. *Communications of the ACM*, 33(5):539 – 543, 1990.

[112] B. N. Karp. *Geographic routing for wireless networks*. PhD thesis, Harvard University, Cambridge, MA, USA, 2000. Adviser-Kung, H. T.

[113] R. M. Karp. Probabilistic analysis of partitioning algorithms for the traveling salesman problem in the plane. *Mathematics of Operations Research*, 2:209 – 224, 1977.

[114] S. Kellner, M. Pink, D. Meier, and E.-O. Blass. Towards a realistic energy model for wireless sensor networks. In *Fifth Annual Conference on Wireless on Demand Network Systems and Services, 2008. WONS 2008.*, pages 97 –100, jan. 2008.

[115] J. Kennedy. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC 1999)*, pages 1931 – 1938, 1999.

[116] J. Kho, A. Rogers, and N. R. Jennings. Decentralized control of adaptive sampling in wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(3):1–35, 2009.

[117] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 4598(220):671–680, May 1983.

[118] J. Knowles and D. Corne. The pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 9 – 105. IEEE Press, 1999.

[119] J. Knowles, L. Thiele, and E. Zitzler. A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. Technical Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, 2006.

[120] A. Konstantinidis, K. Yang, Q. Zhang, and D. Zeinalipour-Yazti. A multi-objective evolutionary algorithm for the deployment and power assignment problem in wireless sensor networks. *Computer Networks*, 54(6):960 – 976, 2010. New Network Paradigms.

[121] S. Kotrotsos, G. Kotsakis, P. Demestichas, E. Tzifa, V. Demesticha, and M. Anagnostou. Formulation and computationally efficient algorithms for an interference-oriented version of the frequency assignment problem. *Wireless Personal Communications*, 18:289 – 317, 2001.

[122] F. Koushanfar, S. Slijepcevic, M. Potkonjak, and A. Sangiovanni-Vincentelli. Location discovery in ad-hoc wireless sensor networks. In X. Cheng, X. Huang, and D.-Z. Du, editors, *Ad Hoc Wireless Networking*, pages 137–173. Kluwer Academic Publishers, 2003.

[123] F. Koushanfar, S. Slijepcevic, J. Wong, and M. Potkonjak. Global error-tolerant algorithms for location discovery in ad-hoc wireless networks. *Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on*, 4:IV–4186 vol.4–, 2002.

[124] J. R. Koza. *Genetic Programming. On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, Massachusetts, 1992.

[125] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 64–75, New York, NY, USA, 2005. ACM.

[126] M. Kuorilehto, M. Hännikäinen, and T. D. Hämäläinen. A survey of application distribution in wireless sensor networks. *EURASIP J. Wirel. Commun. Netw.*, 2005(5):774–788, 2005.

[127] M. Laguna and R. Martí. *Scatter Search. Methodology and Implementations in C*. Kluwer, 2003.

[128] W. Lai and I. C. Paschalidis. Optimally balancing energy consumption versus latency in sensor network routing. *ACM Trans. Sen. Netw.*, 4(4):1–28, 2008.

[129] O. Landsiedel and K. Wehrle. Aeon: Accurate prediction of power consumption in sensor networks. In *In Proceedings of The Second IEEE Workshop on Embedded Networked Sensors (EmNetS-II*, 2004.

[130] K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, page 8 pp., april 2006.

[131] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Optimization by learning and simulation of Bayesian and Gaussian networks. Technical Report KZZA-IK-4-99, Department of Computer Science and Artificial Intelligence, University of the Basque Country, 1999.

[132] L. Lazos and R. Poovendran. Serloc: Robust localization for wireless sensor networks. *ACM Trans. Sen. Netw.*, 1(1):73–100, 2005.

[133] L. Lazos and R. Poovendran. Stochastic coverage in heterogeneous sensor networks. *ACM Trans. Sen. Netw.*, 2(3):325–358, 2006.

[134] L. Lazos, R. Poovendran, and J. A. Ritcey. Analytic evaluation of target detection in heterogeneous wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(2):1–38, 2009.

[135] A. Lédeczi, A. Nádas, P. Völgyesi, G. Balogh, B. Kusy, J. Sallai, G. Pap, S. Dóra, K. Molnár, M. Maróti, and G. Simon. Countersniper system for urban warfare. *ACM Trans. Sen. Netw.*, 1(2):153–177, 2005.

[136] S. Lederer, Y. Wang, and J. Gao. Connectivity-based localization of large-scale sensor networks with complex shape. *ACM Trans. Sen. Netw.*, 5(4):1–32, 2009.

[137] F. L. Lewis. *Smart Environments: Technologies, Protocols, and Applications*, chapter Wireless Sensor Networks, pages 251 – 285. John Wiley, 2004.

[138] L. Li and T. Kunz. Cooperative node localization using nonlinear data projection. *ACM Trans. Sen. Netw.*, 5(1):1–26, 2009.

[139] X. Li, H. Shi, and Y. Shang. A map-growing localization algorithm for ad-hoc wireless sensor networks. In *ICPADS '04: Proceedings of the Parallel and Distributed Systems, Tenth International Conference*, page 395, Washington, DC, USA, 2004. IEEE Computer Society.

[140] X.-Y. Li, W.-Z. Song, and Y. Wang. Localized topology control for heterogeneous wireless sensor networks. *ACM Trans. Sen. Netw.*, 2(1):129–153, 2006.

[141] H. Lim and J. C. Hou. Distributed localization for anisotropic sensor networks. *ACM Trans. Sen. Netw.*, 5(2):1–26, 2009.

[142] D. Liu, P. Ning, and W. Du. Attack-resistant location estimation in sensor networks. *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 99–106, April 2005.

[143] D. Liu, P. Ning, and W. Du. Detecting malicious beacon nodes for secure location discovery in wireless sensor networks. *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 609–619, June 2005.

[144] X. Liu, Q. Wang, W. He, M. Caccamo, and L. Sha. Optimal real-time sampling rate assignment for wireless sensor networks. *ACM Trans. Sen. Netw.*, 2(2):263–295, 2006.

[145] H. R. Lourenço, O. Martin, and T. Stützle. *Handbook of Metaheuristics*, chapter Iterated local search, pages 321 – 353. Kluwer Academic Publishers, 2002.

[146] F. Luna, E. Alba, and A. J. Nebro. Parallel heterogeneous metaheuristics. In E. Alba, editor, *Parallel Metaheuristics*, pages 395 – 422. Wiley, 2005.

[147] F. Luna, A. J. Nebro, and E. Alba. Parallel evolutionary multiobjective optimization. In N. . Nedjah, E. Alba, and L. de Macedo, editors, *Parallel Evolutionary Computations*, volume 22 of *Studies in Computational Intelligence*, chapter 2, pages 33 – 56. Springer, 2006.

[148] G. Luque. *Resolución de Problemas Combinatorios con Aplicacin Real en Sistemas Distribuidos*. PhD thesis, University of Málaga, 2006.

[149] Y. Ma and J. H. Aylor. System lifetime optimization for heterogeneous sensor networks with a hub-spoke topology. *IEEE Transactions on Mobile Computing*, 3(3):286–294, 2004.

[150] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 88–97, 2002.

[151] P. Manohar, S. S. Ram, and D. Manjunath. Path coverage by a sensor field: The nonhomogeneous case. *ACM Trans. Sen. Netw.*, 5(2):1–26, 2009.

[152] C. Maple, L. Guo, and J. Zhang. Parallel genetic algorithms for third generation mobile network planning. In *Proceedings of the International Conference on Parallel Computing in Electrical Engineering (PARELEC04)*, pages 229–236, 2004.

[153] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless ad-hoc sensor networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 139–150, New York, NY, USA, 2001. ACM.

[154] Z. Meng, S. Wang, and Q. Wang. Fault tolerant topology control for clustered wireless sensor networks. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, pages 1 –5, oct. 2008.

[155] G. Merrett, A. Weddell, A. Lewis, N. Harris, B. Al-Hashimi, and N. White. An empirical energy model for supercapacitor powered wireless sensor nodes. In *Proceedings of 17th International Conference on Computer Communications and Networks, 2008. ICCCN '08.*, pages 1 –6, aug. 2008.

[156] W. Merrill, F. Newberg, L. Girod, and K. Sohrabi. Battlefield ad-hoc lans: a distributed processing perspective. *GOMACTech*, 2004.

[157] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087 – 1092, 1953.

[158] H. Meunier, E.-G. Talbi, and P. Reininger. A multiobjective genetic algorithm for radio network optimization. *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, 1:317–324 vol.1, 2000.

[159] F. Mingyue, Y. Xianqing, L. Guohui, D. Zhanshuai, and W. Xiangneng. Sensor scheduling for target tracking in a wireless sensor network using modified particle swarm optimization. In *ISCSCT '08: Proceedings of the 2008 International Symposium on Computer Science and Computational Technology*, pages 156–159, Washington, DC, USA, 2008. IEEE Computer Society.

[160] A. R. Mishra. *Fundamentals of Cellular Network Planning and Optimisation: 2G/2.5G/3G... Evolution to 4G*. Wiley, 2004.

[161] N. Mladenovic and P. Hansen. Variable neighborhood search. *Com. Oper. Res*, 24:1097 – 1100, 1997.

[162] N. Mladineo and S. Knezic. Optimisation of forest fire sensor network using gis technology. *Information Technology Interfaces, 2000. ITI 2000. Proceedings of the 22nd International Conference on*, pages 391–396, 13-16 June 2000.

[163] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 50–61, New York, NY, USA, 2004. ACM.

[164] P. Moscato. Memetic algorithms: A short introduction. In *New ideas in optimization*, pages 219–234, Maidenhead, UK, 1999. McGraw-Hill Ltd.

[165] H. Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5:303 – 346, 1998.

[166] G. Nan and M. Li. Evolutionary based approaches in wireless sensor networks: A survey. In *Fourth International Conference on Natural Computation, 2008. ICNC '08.*, volume 5, pages 217 –222, oct. 2008.

[167] A. Nasipuri and K. Li. A directionality based location discovery scheme for wireless sensor networks. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 105–111, New York, NY, USA, 2002. ACM.

[168] A. J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J. J. Durillo, and A. Beham. Abyss: Adapting scatter search to multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 12(4), August 2008.

[169] J. Nemeroff, L. Garcia, D. Hampel, and S. DiPierro. Application of sensor network communications. *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, 1:336–341 vol.1, 2001.

[170] D. Niculescu and B. Nath. Ad hoc positioning system (aps). In *Global Telecommunications Conference, 2001. GLOBECOM 2001. IEEE*, pages 2926–2931, 2001.

[171] D. Niculescu and B. Nath. Error characteristics of ad hoc positioning systems (aps). In *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 20–30, New York, NY, USA, 2004. ACM.

[172] A. Osyczka. Multicriteria optimization for engineering design. In J. S. Gero, editor, *Design Optimization*, pages 193 – 227. Academic Press, 1895.

[173] V. Pareto. *Cours D'Economie Politique*, volume I and II. F. Rouge, Lausanne, 1896.

[174] I. C. Paschalidis and D. Guo. Robust and distributed stochastic localization in sensor networks: Theory and experimental results. *ACM Trans. Sen. Netw.*, 5(4):1–22, 2009.

[175] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. BOA: The Bayesian optimization algorithm. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, volume 1, pages 525 – 532. Morgan Kaufmann Publishers, San Francisco, CA, 1999.

[176] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *IEEE WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS*, pages 90–100, 1999.

[177] P. Pradhan, V. Baghel, G. Panda, and M. Bernard. Energy efficient layout for a wireless sensor network using multi-objective particle swarm optimization. In *Advance Computing Conference, 2009. IACC 2009. IEEE International*, pages 65 –70, 6-7 2009.

[178] C. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publishing, Oxford, UK, 1993.

[179] P. Reininger and A. Caminada. Model for GSM radio network optimization. In *Second ACM International Conference on Discrete Algorithms and Methods for Mobility*, 1998.

[180] P. Reininger and A. Caminada. Multicriteria design model for cellular network. *Annals of Operations Research*, 107:251 – 265, 2001.

[181] J. Sarma and K. D. Jong. An Analysis of Local Selection Algorithms in a Spatially Structured Evolutionary Algorithm. In T. Bäck, editor, *Proceedings of the 7th International Conference on Genetic Algorithms*, pages 181–186. Morgan Kaufmann, 1997.

[182] A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 166–179, New York, NY, USA, 2001. ACM.

[183] L. Schwiebert, S. K. Gupta, and J. Weinmann. Research challenges in wireless networks of biomedical sensors. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 151–165, New York, NY, USA, 2001. ACM.

[184] C. Sengul, M. J. Miller, and I. Gupta. Adaptive probability-based broadcast forwarding in energy-saving sensor networks. *ACM Trans. Sen. Netw.*, 4(2):1–32, 2008.

[185] D. Shamsi, F. Koushanfar, and M. Potkonjak. Challenging benchmark for location discovery in ad hoc networks: foundations and applications. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 361–370, New York, NY, USA, 2008. ACM.

[186] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from connectivity in sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 15:961–974, 2004.

[187] D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. CRC Press, 2003.

[188] Y. Shi and Y. T. Hou. Optimal base station placement in wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(4):1–24, 2009.

[189] V. Shnayder, M. Hempstead, B. rong Chen, G. W. Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 188–200, New York, NY, USA, 2004. ACM.

[190] N. Shrivastava, R. Mudumbai, U. Madhow, and S. Suri. Target tracking with binary proximity sensors. *ACM Trans. Sen. Netw.*, 5(4):1–33, 2009.

[191] N. Shrivastava, S. Suri, and C. D. Tóth. Detecting cuts in sensor networks. *ACM Trans. Sen. Netw.*, 4(2):1–25, 2008.

[192] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 181–190, Dallas, Texas, 1998. ACM.

[193] B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert, and S. S. Sastry. Distributed control applications within sensor networks. In *IEEE Proceedings Special Issue on Distributed Sensor Networks*, pages 1235–1246, 2003.

[194] S. Slijepcevic, S. Megerian, and M. Potkonjak. Location errors in wireless embedded sensor networks: sources, models, and effects on applications. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(3):67–78, 2002.

[195] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. *Communications, 2001. ICC 2001. IEEE International Conference on*, 2:472–476, 2001.

[196] M. Soto, A. Ochoa, S. Acid, and L. M. de Campos. Introducing the polytree aproximation of distribution algorithm. In *Second Symposium on Artificial Intelligence. Adaptive Systems. CIMAF 99*, pages 360 – 367, 1999.

[197] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221 – 248, 1994.

[198] T. Stützle. Iterated local search for the quadratic assignment problem. Technical Report aida-99-03, FG Intellektik, TU Darmstadt, 1999.

[199] X. Su. A combinatorial algorithmic approach to energy efficient information collection in wireless sensor networks. *ACM Trans. Sen. Netw.*, 3(1):6, 2007.

[200] K. Sun, P. Ning, and C. Wang. Fault-tolerant cluster-wise clock synchronization for wireless sensor networks. *Dependable and Secure Computing, IEEE Transactions on*, 2(3):177 – 189, july-sept. 2005.

[201] A. Swain and R. Hansdah. An energy efficient and fault-tolerant clock synchronization protocol for wireless sensor networks. In *Communication Systems and Networks (COMSNETS), 2010 Second International Conference on*, pages 1 –10, jan. 2010.

[202] M. T. Thai, F. Wang, D. H. Du, and X. Jia. Coverage problems in wireless sensor networks: designs and analysis. *Int. J. Sen. Netw.*, 3(3):191–200, 2008.

[203] A. Tiwari, P. Ballal, and F. L. Lewis. Energy-efficient wireless sensor network design and implementation for condition-based maintenance. *ACM Trans. Sen. Netw.*, 3(1):1, 2007.

[204] Y.-C. Tseng, C.-F. Huang, and S.-P. Kuo. Positioning and Location Tracking in Wireless Sensor Networks. In M. Ilyas and I. Mahgoub, editors, *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. Kluwer Academic Publishers, 2005.

[205] D. A. V. Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis. Technical Report TR-98-03, Dept. Elec. Comput. Eng., Graduate School of Eng., Air Force Inst.Technol., Wright-Patterson, AFB, OH, 1998.

[206] G. Veltri, Q. Huang, G. Qu, and M. Potkonjak. Minimal and maximal exposure path algorithms for wireless embedded sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 40–50, New York, NY, USA, 2003. ACM.

[207] M. A. M. Vieira, L. F. M. Vieira, L. B. Ruiz, A. A. F. Loureiro, A. O. Fernandes, and J. M. S. Nogueira. Scheduling nodes in wireless sensor networks: A voronoi approach. In *LCN '03: Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks*, page 423, Washington, DC, USA, 2003. IEEE Computer Society.

[208] C. Wang and L. Xiao. Sensor localization in concave environments. *ACM Trans. Sen. Netw.*, 4(1):1–31, 2008.

[209] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 28–39, New York, NY, USA, 2003. ACM.

[210] S. Watanabe, T. Hiroyasu, and M. Mikiand. Parallel evolutionary multi-criterion optimization for mobile telecommunication networks optimization. In *Proceedings of the EUROGEN2001 Conference*, pages 167–172, Athens, Greece, September 19-21, 2001.

[211] A. S. Weddell, N. R. Harris, and N. M. White. Alternative energy sources for sensor nodes: Rationalized design for long-term deployment. In *Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008. IEEE*, pages 1370 –1375, may 2008.

[212] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, 2006.

[213] T. A. Wettergren and R. Costa. Optimal placement of distributed sensors against moving targets. *ACM Trans. Sen. Netw.*, 5(3):1–25, 2009.

[214] J. Whittaker. *Graphical models in applied multivariate statistics*. John Wiley & Sons, Inc., 1990.

[215] M. Woehrle, D. Brockhoff, T. Hohm, and S. Bleuler. Investigating Coverage and Connectivity Tradeoffs in Wireless Sensor Networks: The Benefits of MOEAs. In M. Ehrgott et al., editors, *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems (MCDM 2008)*, volume 634 of *LNEMS*, pages 211–221, Heidelberg, Germany, 2010. Springer.

[216] J. L. Wong, R. Jafari, and M. Potkonjak. Gateway placement for latency and energy efficient data aggregation [wireless sensor networks]. In *29th Annual IEEE International Conference on Local Computer Networks, 2004.*, pages 490 – 497, nov. 2004.

[217] J. Wu and S. Yang. Coverage issue in sensor networks with adjustable ranges. In *Proceedings. 2004 International Conference on Parallel Processing Workshops, 2004. ICPP 2004 Workshops.*, pages 61 – 68, aug. 2004.

[218] Q. Wu, N. S. V. Rao, X. Du, S. S. Iyengar, and V. K. Vaishnavi. On efficient deployment of sensors on planar grid. *Comput. Commun.*, 30(14-15):2721–2734, 2007.

[219] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity conguration for energy conservation in sensor networks. *ACM Trans. Sen. Netw.*, 1(1):36–72, 2005.

[220] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 13–24, New York, NY, USA, 2004. ACM.

[221] W. Xu, W. Trappe, and Y. Zhang. Defending wireless sensor networks from radio interference through channel adaptation. *ACM Trans. Sen. Netw.*, 4(4):1–34, 2008.

[222] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 70–84, New York, NY, USA, 2001. ACM.

[223] Y. Xu and X. Yao. A ga approach to the optimal placement of sensors in wireless sensor networks with obstacles and preferences. In *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, volume 1, pages 127 – 131, 8-10 2006.

[224] T. Yan, T. He, and J. A. Stankovic. Differentiated surveillance for sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 51–62, New York, NY, USA, 2003. ACM.

[225] K.-K. Yap, V. Srinivasan, and M. Motani. Max: Wide area human-centric search of the physical world. *ACM Trans. Sen. Netw.*, 4(4):1–34, 2008.

[226] M. Younis and K. Akkaya. Strategies and techniques for node placement in wireless sensor networks: A survey. *Ad Hoc Netw.*, 6(4):621–655, 2008.

[227] M. R. Yuce, S. W. Ng, N. L. Myo, J. Y. Khan, and W. Liu. Wireless body sensor network using medical implant band. *J. Med. Syst.*, 31(6):467–474, 2007.

[228] M. Z. Zamalloa, K. Seada, B. Krishnamachari, and A. Helmy. Efficient geographic routing over lossy links in wireless sensor networks. *ACM Trans. Sen. Netw.*, 4(3):1–33, 2008.

[229] H. Zhang and J. C. Hou. On the upper bound of α-lifetime for large sensor networks. *ACM Trans. Sen. Netw.*, 1(2):272–300, 2005.

[230] X. Zhang and S. B. Wicker. On the optimal distribution of sensors in a random field. *ACM Trans. Sen. Netw.*, 1(2):301–306, 2005.

[231] Y. Zheng, D. J. Brady, and P. K. Agarwal. Localization using boundary sensors: An analysis based on graph theory. *ACM Trans. Sen. Netw.*, 3(4):21, 2007.

[232] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang. Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In *2006 IEEE Congress on Evolutionary Computation*, pages 3234–3241, 2006.

[233] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Models and solutions for radio irregularity in wireless sensor networks. *ACM Trans. Sen. Netw.*, 2(2):221–262, 2006.

[234] Z. Zhou, S. R. Das, and H. Gupta. Variable radii connected sensor cover in sensor networks. *ACM Trans. Sen. Netw.*, 5(1):1–36, 2009.

[235] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithms. In K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, editors, *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2002. International Center for Numerical Methods in Engineering (CIMNE).

[236] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.