Towards Fault Reactiveness in Wireless Sensor Networks with Mobile Carrier Robots

by

Rafael Jesus Falcon Martinez

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science

School of Electrical Engineering and Computer Science

Faculty of Engineering

University of Ottawa

"*If I have seen farther than others,*

it is by standing upon the shoulders of giants."

– Sir Isaac Newton

ABSTRACT

Wireless sensor networks (WSN) increasingly permeate modern societies nowadays. But in spite of their plethora of successful applications, WSN are often unable to surmount many operational challenges that unexpectedly arise during their lifetime. Fortunately, robotic agents can now assist a WSN in various ways. This thesis illustrates how mobile robots which are able to carry a limited number of sensors can help the network react to sensor faults, either during or after its deployment in the monitoring region.

Two scenarios are envisioned. In the first one, carrier robots surround a point of interest with multiple sensor layers (*focused coverage formation*). We put forward the first known algorithm of its kind in literature. It is energy-efficient, fault-reactive and aware of the bounded robot cargo capacity. The second one is that of replacing damaged sensing units with spare, functional ones (*coverage repair*), which gives rise to the formulation of two novel combinatorial optimization problems. Three nature-inspired metaheuristic approaches that run at a centralized location are proposed. They are able to find good-quality solutions in a short time.

Two frameworks for the identification of the damaged nodes are considered. The first one leans upon *diagnosable systems*, i.e. existing distributed detection models in which individual units perform tests upon each other. Two swarm intelligence algorithms are designed to quickly and reliably spot faulty sensors in this context. The second one is an *evolving risk management framework* for WSNs that is entirely formulated in this thesis.

ACKNOWLEDGMENTS

My deepest gratitude to my thesis supervisor, Dr. Amiya Nayak and my thesis cosupervisor, Dr. Ivan Stojmenovic, for their brilliant scientific guidance and steady financial support throughout the doctoral program. They have challenged me in many ways and made me a stronger researcher.

I would also like to thank a few talented colleagues with whom I have had the privilege to collaborate during these years: Dr. Xu Li from INRIA Lille-Nord Europe (France), Dr. Benoît Depaire, Dr. An Caris and Dr. Koen Vanhoof from Hasselt Universiteit (Belgium), Dr. Marcio Almeida from the University of Ottawa (Canada) and Dr. Rami Abielmona from Larus Technologies Corporation (Canada). It has been a great privilege to work alongside you and witness your exceptional research skills.

Special thanks go to Dr. Rafael Bello, Dr. María M. García, Dr. Yanet Rodríguez and Dr. Miriam Nicado at the Universidad Central de Las Villas (Cuba), with whom I learned to walk in science. They introduced me to the fascinating world of Computational Intelligence.

Finally, I want to sincerely thank my parents, my girlfriend Megan and my brothers Orly and David for their continuous support and encouragement.

This work was partially supported by NSERC Strategic Grants from the thesis supervisors as well as by Canada's IRAP and MITACS Accelerate programs, Larus Technologies Corporation and the IEEE CIS "Walter Karplus" Research Grant awarded to this candidate.

CONTENTS

FIGURES

TABLES

CHAPTER 1

INTRODUCTION

The aim of this introductory chapter is to plunge the reader into the fascinating world of wireless sensor networks and their integration with multi-robot systems, a recent technological breakthrough that is increasingly permeating modern societies nowadays. We unveil the potential benefits of **exploiting carrier robots for achieving fault-reactive topologies in wireless sensor networks**, a promising yet rather untrodden topic that stands as the major driving force of this doctoral study.

After an initial motivation and clearly stating the overall research goal and its constituent milestones, we present the list of algorithmic contributions that support the scientific novelty of the thesis. The chapter concludes with an outline of the manuscript structure.

1.1 Motivation

As scientific research and technological innovation rapidly advance, humans become familiar with newer kinds of artifacts that smooth daily activities and thus threaten to take a firm grip on both individual and corporate well being. The ubiquity of traditional computers is slowly receding to the massive influx of smart phones, tablets, global positioning systems (GPS) and cloud computing devices. Sensors are oftentimes an integral part of such cutting-

edge systems for their ability to measure a wide variety of indicators, from temperature and motion to pressure and light intensity.

It is in the collaborative interaction of standalone sensing units where greater potential lies and research studies focus on. A *wireless sensor network* (WSN) [130] is a collection of autonomous sensing nodes that communicate via wireless links. They are deployed in indoor and outdoor scenarios alike and are expected to work unattended monitoring the region and reporting their measurements to a central location.

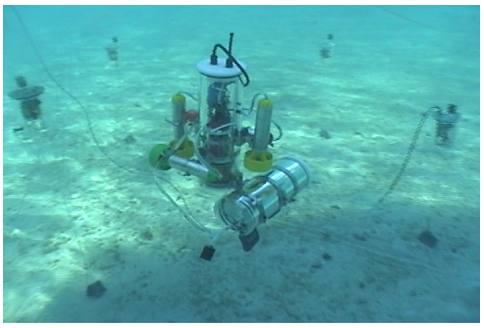
In spite of their large chain of successful applications in dissimilar domains [16, 67, 68, 135], WSNs are often unable to surmount many operational challenges that unexpectedly arise during their lifetime [64], such as resource depletion (energy, memory, processing), connectivity disruption, malicious attacks or harsh environmental conditions.

The task of assisting WSNs in manifold aspects has been recently entrusted to robotic agents. The latest advances in the field of multi-robot systems have made possible to seamlessly integrate robotic and sensory devices into a paradigmatic class of cooperative networking scenarios coined as *wireless sensor and robot network* (WSRN). A WSRN usually involves resource-rich, mobile robots tending resource-constrained, stationary sensors. Actions are collaboratively executed by the robotic agents on the basis of the information received by the deployed, standalone sensing units. A WSRN could be thought of as a distributed control system that needs to timely react to sensor information with an appropriate action [102].

Two ongoing research projects on WSRNs undertaken by the Distributed Robotics Lab (DRL) at the Massachusetts Institute of Technology (MIT) will help illustrate the potential unleashed by mobile robots in boosting the performance of a sensor network.

The first scenario is concerned with an underwater WSRN [41] in which nodes are networked optically, acoustically and using radio. The sensor units (Aquanodes) in Fig. 1.1(a) are equipped with communication, sensing and processing modules and packed in cylindrical watertight containers. Sensors localize static and moving objects via one-way ranging provided by a built-in acoustic modem. They measure water temperature, pressure and other

(a) Aquanodes drying.



(b) The network deployed in Moorea, French Polynesia.

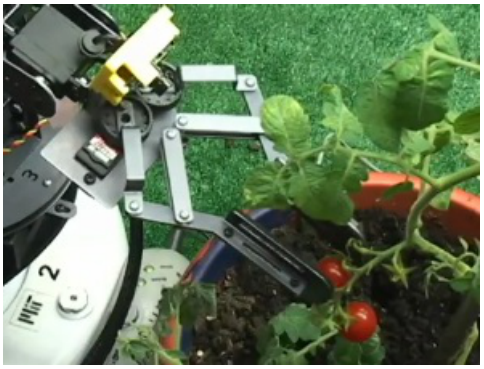
Figure 1.1: The AMOUR underwater WSRN. © DRL, MIT.

chemical indicators. The Aquanodes are anchored with weights and thus form a static sensor network. Mobile nodes are of Autonomous Modular Optical Underwater Robot (AMOUR) type and they have the functionality of the Aquanodes plus a more advanced camera system to avoid obstacles. AMOUR units aim at traveling around and downloading data from the sensors, as shown in Fig. 1.1(b). Additionally, if an event of interest is reported, the robots move there and sample the environment as well, thus enhancing the dynamic sampling ability of the network. Experiments have been conducted in oceans, rivers and lakes with promising results.

The second project that captured our attention is the design of a robotic gardening system [33]. The long-term goal is the development of an autonomous greenhouse where mobile robots perform as gardeners by delivering water and nutrients to the plants (as in Fig. 1.2(a)) and harvesting the fruits upon demand (as portrayed in Fig. 1.2(b)). The plants are cherry tomatoes grown in pots endowed with soil sensors that communicate with the robots. The robotic agents possess a custom watering system, arm, eye-in-hand camera and mesh networking. While sensory nodes monitor their soil humidity and issue watering requests, robots are able to sprinkle water and nutrients on a specific plant and locate and grasp a tomato. They use an embedded plant-specific growth model to make predictions about the state of the fruit and thus harvest the ripest tomatoes in response to a user notification.

The aforementioned research projects are good representatives of the two major special-

(a) Robots tending to cherry tomato plants in the greenhouse.



(b) A robot grasping a tomato upon request.

Figure 1.2: The robotic garden WSRN. © DRL, MIT.

ization branches that prevail nowadays in the WSRN arena. The AMOUR underwater system typifies a *robot-assisted WSN* (RA-WSN). In this collaborative framework, the robots are not considered an integral part of the WSN but their goal is to enhance and augment its functionality. In absence of the robotic agents, the WSN remains still operative yet undergoes a mounting performance degradation over time. On the other hand, the robotic gardening system exhibits a more tightly woven fabric of robots and sensors, for the former carry out actions not necessarily to improve the robustness of the latter but as part of a fine-grained, synergetic original design. Removal of the robotic agents will render the WSN completely unable to accomplish its primary purpose. The terms "WSRN" (in the narrow sense) and *wireless sensor and actuator/actor networks* (WSANs) have been indistinctly employed in the literature to describe this sort of cooperative networking system. In this thesis however, we will refer to the second scenario as a *robot-dependent WSN* (RD-WSN) in order to emphasize the high degree of integration of the robotic nodes into the WSN's core functionality. Hence, we keep the term WSRN general enough to embrace both RA- and RD- types of WSNs and in this way avoid any confusion on the role played by the mobile robots in relation to the networked sensor ensemble.

One of the most critical challenges that WSNs face along their operational lifetime is preserving their monitoring capabilities and overall sampling reliability in presence of defective

sensors. It is known that sensing devices are massively manufactured nowadays because of their simple yet fairly feeble architecture. They can become inoperative for a broad array of reasons and in such cases an appropriate response of the WSN is required. Detecting the faulty nodes and taking prompt action is pivotal to ensure the stability of the networked ensemble, whether it is threatened by an obvious sensor shutdown caused by battery depletion or by a more subtle deviation in the accuracy of its reported measurements. While *fault-tolerant* protocols have been the prevalent mainstream in literature for quite some time (i.e. algorithms that allow the network to continue operating in spite of the sensor failure, e.g. by constructing a bi-connected topology that guarantees alternative message routing [22]), they are unable to restore the loss of coverage (area under supervision of the WSN) or surveillance integrity associated with the damaged sensor.

This is one of the major reasons why we are witnessing a renewed interest from the scientific community in *fault-reactive* sensor networks, i.e. groups of interconnected sensing units that dynamically adjust the network topology in response to the identification of defective nodes. A large body of research has been devoted to *mobile sensor networks* (MSNs), where each node is assumed to be mounted on a platform that provides locomotion around the deployment field. In this way, functional sensors can relocate themselves to the coordinates of their out-of-order peers [89]. The problem here is that MSNs' mobility requirement is often unrealistic, especially for large-scale WSNs, given that the ensuing hardware expenditures are likely far beyond the available budget. Instead, a few *mobile carrier robots* (i.e. agents that are able to carry a limited number of sensors) can collaboratively replace damaged, static sensors with working ones as sudden failures emerge. *Robot-based sensor relocation* is a relatively recent topic yet most of the existing protocols wrongly rely on unbounded robot cargo capacity.

The aforementioned limitations have spurred us to enunciate the ultimate aim of this thesis: **to have mobile carrier robots efficiently cooperate with one another in order to achieve fault reactiveness in a stationary WSN**.

We envision a RD-WSN scenario with a robotic carrier fleet surrounding a geographical point of interest with multiple sensor layers (*focused coverage formation*), catering for inoperative sensing units and returning to predefined base points for sensor reloading, as will be discussed in Chapter 3. We also envisage a RA-WSN setting in Chapters 4 and 5, with robots departing from a base station and following nearly-optimal replacement trajectories to pickup passive sensors all over the field and drop them at the location of the defective nodes, then returning to the base station.

Determining which sensors are not behaving properly is an important preliminary step for the two collaborative scenarios described above. Two frameworks for the identification of the damaged nodes are considered. Chapter 6 makes use of a widely popular distributed fault diagnosis framework in which sensors mutually test each other and submit the test outcomes to a supervisory unit. The status of each sensing node (faulty or fault-free) is to be learned from the collection of all tests carried out system-wide. Chapter 6 unveils two swarm intelligence algorithms that quickly and reliably derive the true set of faulty units and also lower the time and memory requirements of the underlying search process, thus making fault identification an inexpensive approach that can be eventually executed by resource-constrained devices (e.g. cluster-head nodes). Then, Chapter 7 introduces an evolving risk management framework for WSNs in which the crisp definition of faulty/fault-free node is enriched with a numerical quantification of the overall risk level posed by any sensing unit, thus resulting in a more flexible and human-centric alternative for fault detection.

The next section states the general thesis goal and its constituent milestones.

1.2 Thesis Goal: Achieving Fault-Reactive Wireless Sensor Networks through Mobile Carrier Robots

The motivational background in the previous section was meant to highlight two core issues: that robotic agents have the potential to optimize many operational metrics of a WSN,

and that faulty sensors pose a major threat to the trustworthiness of its monitoring activities. These two facts are intermingled in the formulation of the overall thesis goal.

"Upon efficient and reliable identification of defective static sensors, either during or after the network deployment, their coordinated replacement with spare, functional units undertaken by mobile carrier robots leads to a fault-reactive wireless sensor network, thus preserving its surveillance integrity."

The building blocks of the general thesis objective (i.e., fault detection and robot-based sensor replacement) have independently been the target of numerous research studies. Both can be attained in a variety of ways, depending on the underlying set of assumptions bound to the scenario under discussion. This thesis provides a unified framework in which both paradigms can be integrated for the sake of fault reactiveness in WSNs. It centers around a selected group of case studies with practical relevance to modern life, for which problems are enunciated and solution methods are put forward. The present research work is not meant to serve as a universal, generalizable body of theoretical knowledge and pragmatic solutions when it comes to fault-reactive WSNs. Rather, it intends to spark further research endeavours from academia and industry in the pursuit of sensing environments with higher degrees of autonomy and reliability.

For a better understanding of the explored research avenues, the overall goal is split into three milestones. We first assume that all defective sensors have been identified and elaborate on the coverage formation and repair protocols followed by the robot(s) to restore the network integrity. Then, we take a closer look at the fault diagnosis process under two different frameworks.

1.2.1 Milestone 1: Carrier-based Focused Coverage Formation

The challenging task of deploying a sensor network in environments where human intervention is impossible or bears scant likelihood to succeed (e.g. hazardous or inaccessible terrains) relies to an increasing degree in the abilities of robotic swarms. In the realm of robot-

based sensor placement (i.e. mobile actuators laying sensors at desired coordinates), the vast majority of the published papers target the so-called "*blanket coverage*" problem, in which the objective is to drop sensors in such a way that the total area under network surveillance is maximized. An overwhelming number of such articles unrealistically assume that robots have unlimited cargo capacity, thus making their algorithms unfit for real-life applications.

Recently, a novel type of coverage problem in WSN was proposed in [87]. It was coined as "*focused coverage*" because the intent is to surround a geographical point of interest (POI) with multiple sensor layers. The closer we get to the POI, the more important it is to provide reliable surveillance, as is the case in a plethora of post-disaster management scenarios where, for instance, buildings partially hit by earthquakes or tsunami waves must be carefully monitored to prevent their collapse and save valuable human lives.

The goal is to have robots cooperate among themselves and with sensors in order to construct a networking layout around the POI that maximizes its distance to uncovered areas, while keeping an eye on the total distance traveled by the robotic nodes. Additionally, the entailed communication overhead must be minimized, obstacles avoided and defective sensors substituted with spare ones all along, i.e. during or after the network deployment process.

In Chapter 3, we introduce in detail the first approach that, to the best of our knowledge, utilizes a robotic carrier fleet to construct a focused coverage formation. The algorithmic design takes into account the bounded robot cargo capacity and satisfies the above demands. Because robots play a pivotal role during the WSN formation, this novel cooperative networking scenario can be categorized as RD-WSN.

1.2.2 Milestone 2: Carrier-based Coverage Repair

Consider the setting in which the WSN has been deployed somehow and one or more mobile carrier robots lie at a central location called "base station", where data from every single sensing unit are received. Because of the abundance of cheap sensors scattered, not all of them are actually needed to provide area coverage. Hence, they follow a scheduling

algorithm to decide which sensors will go into sleep mode (*passive units*) and which will monitor the region (*active units*). Unfortunately, the quality of the network coverage will be eventually degraded due to failures of the active units because of e.g., battery depletion, unexpected events, hardware or software faults, etc.

The mission of the robotic team is to collect passive nodes all over the field and drop them at the positions of the faulty sensors. A corporate *route plan* (i.e. set of individual routes departing from and returning to the base station) for robots to follow is to be derived. The purpose is to minimize the cost of the coverage repair operation (in terms of e.g. distance traveled, replacement latency, etc.) while ensuring that the network remains responsive, i.e. there is little time to find a solution given that an action must be taken swiftly.

The previous problem is novel in literature. We baptized it as *carrier-based coverage-repair* (CBCR) and cast it into the combinatorial optimization world. Due to its underlying intricacy, approximate solutions of good quality are sought. In Chapter 4, we study the single-robot case and put forth two nature-inspired metaheuristic schemes that elegantly solve it. Chapter 5 focuses on the multiple-robot variant and presents a hybrid metaheuristic as the first known solution approach for this problem. Given the auxiliary nature of the team of mobile robots and the fact that the WSN could still be operative in its absence, the proposed CBCR setting falls under the umbrella of RA-WSN cooperative networking systems.

1.2.3 Milestone 3: Identification of the Defective Sensing Units

The coverage formation and repair scenarios described in milestones 1.2.1 and 1.2.2 are contingent upon the identification of the defective nodes. The fault diagnosis process in a WSN must yield reliable results (i.e. there must be high certainty on why the predicted nodes are truly damaged, otherwise the operational perils faced by the system are not removed) and run in an efficient manner (time to perform the diagnosis is usually limited).

Two frameworks for the discovery of the damaged nodes in a WSN are considered. The first one is referred to as *diagnosable systems* [65] and is described in Chapter 6. A parallel or

distributed system is regarded as diagnosable if its individual units test each other and submit the verification result to a supervisory node for further analysis. The test outcome is of binary nature and reflects the belief that the tested node is faulty (1) or fault-free (0). Nodes are allowed to carry out verifications upon their peers regardless of what their condition is. Tests executed by fault-free nodes are deemed reliable whereas those conducted by faulty units are not. The task of deriving the true node status out of the collection of tests carried out system-wide is a well-studied problem referred to as *system-level fault diagnosis* [58]. When posed as an optimization problem under arbitrary network topologies, it bears non-polynomial complexity and hence calls for the introduction of approximate algorithms to solve medium-sized and large system instances.

To guarantee the veracity of the fault prediction (and thus remove any concern on whether nodes are being wrongly labeled as faulty or fault-free), we adopt two popular test models (invalidation and comparison) in presence of a kind of diagnosable systems in which the connection assignment among nodes has been carefully crafted to ensure a deterministic prediction outcome. Then in Chapter 6, two scalable swarm intelligence algorithms that quickly locate the true group of faulty units for any network topology are thoroughly unfolded.

The distributed fault detection framework in Chapter 6 could be complemented with a more human-centric counterpart. In diagnosable systems, individual sensor nodes are responsible for locally conducting the testing phase while the identification of the faulty nodes is entrusted to a supervisory unit. This process occurs in a highly automated fashion once the manifold risk factors that a sensing node is exposed to (e.g. battery level, proximity to a hazardous source, inconsistency of the measurements over time, etc.) have been hardwired a priori into the sensor's architecture. A time-evolvable risk perception is hard to assimilate by such systems and the communication overhead entailed by such an update is non-negligible. Additionally, system-level fault diagnosis forces the risk assessment of any sensor to take a binary form (faulty or fault-free), thus leaving enough ground for concerns on the extent of the risk posed by the sensing unit at any point in time. Finally, upon receiving the set of all test

outcomes from the field, the supervisory unit must initiate a search process in order to determine the true label (status) of each node in the WSN if no topological knowledge is exploited. This search process could be time-consuming depending on the size of the WSN.

While automation in the fault identification task continues to be necessary so that the WSRN can still work unattendedly, we would like the system to gladly override its understanding of risk (either for a particular sensor or globally) with that provided by a human expert who is occasionally monitoring it. There is little work in the literature when it comes to *risk management frameworks for WSNs* and our expectation is to narrow that gap. Chapter 7 introduces an evolving version of a risk management framework featuring a multi-modular architectural design. One of its distinctive features is the quantification of the level of risk attached to any sensing unit, which provides a smoother and superior treatment of the "damaged" units that serve as input to the robot-operated WSN coverage formation and repair protocols enunciated in the first two milestones. In other words, the carrier robots will replace the "riskiest" sensors in the deployment field, thus making the WSN fault-reactive.

1.3 Contributions

The scientific novelty behind this doctoral thesis comes in the form of the following contributions:

1.3.1 Carrier-based Focused Coverage Formation

To the best of our knowledge, we are the first ones formalizing a focused coverage formation (FCF) problem in which the sensor placement is carried out by mobile actuators (robots). FCF emerged quite recently [87] and previous studies [88, 90] have all focused on mobile sensor networks, with the consequent waste of network design budget (owing to the purchase of mobile node platforms) and precious locomotion power during the algorithms' execution.

Chapter 3 unveils our solution protocol, termed *Carrier-based Focused Coverage Forma-*

tion (CBFCF). Robots enter the environment from fixed locations, called "base points", and move toward the POI. As soon as they get in touch with already deployed sensors, they search (by communication instead of by mobility, which is several orders of magnitude more expensive) along the network border for best sensor placement spots (to improve the focused coverage) and move to drop sensors at the discovered locations. Border nodes store the coordinates of failed sensors (if any exists) inside the network as well as of adjacent available deployment positions outside the network, and recommend them to robots during the search process. Robots return to base points for reloading after deploying their current payload and immediately re-enter the environment to augment existing F-coverage. An optimization technique was incorporated so as to reduce augmentation delay and save robot energy.

Several properties turn CBFCF into an appealing choice for real-time deployment scenarios. It is strictly localized (meaning that the communication overhead is limited to the immediate neighborhood of the participating sensors and robots), obstacle-aware, efficient in its inter-robot coordination mechanisms, guarantees maximal coverage in absence of sensor failures and remains robust when they occur. Moreover, it is realistic about the cargo limitations of carrier agents and yields a bi-connected network layout around the POI.

Analytical insights on the correctness and reliability of CBFCF, together with a comprehensive simulation study conducted to assess its performance in terms of energy expenditures and deployment latency, complete the description in Chapter 3 of the proposed carrier-based FCF scheme. The research endeavours in this area led to the publication of [50] and [51].

1.3.2 Carrier-based Coverage Repair

We pioneer also in the CBCR arena by first envisioning the idea of robot-based sensor relocation with an underlying wake-up scheduling algorithm and multiple robotic units lying at a base station. This is a real-world coverage repair scenario stemming in RA-WSNs and of practical relevance for the development of self-healing autonomous sensing environments. The solo-robot CBCR variant (coined as *single-carrier coverage repair*, SC^2R) was

formulated as a particular instance of a new combinatorial optimization problem (*the one-commodity traveling salesman problem with selective pickup and delivery*, 1-TSP-SELPD). SC^2R is a special case of 1-TSP-SELPD in which any node's demand is restricted to exactly one unit. Nearly-optimal, approximate solutions to SC^2R were generated via population-based metaheuristic approaches that draw inspiration from biological and social systems.

1-TSP-SELPD turned out to be a generalization of the Traveling Salesman Problem (TSP), a true landmark in combinatorial optimization. It borrows elements from two TSP extensions (Pickup-and-Delivery TSP and Prize-Collecting TSP) yet stands on its own as a still unexplored variant in literature. Compared to their TSP-based parents, a solution to the novel problem (robot's tour) does not require the visitation of all graph vertices. This is advantageous on the one hand because the dimensionality of the search space is reduced. However, there is an extra layer of complexity: figuring out which nodes must be included in the tour and what their right permutation order is. This makes good solutions harder to find and, together with the limited time usually available for computing them (due to the inherent responsiveness of WSRNs), pose a serious challenge to the performance of any algorithmic solution. In Chapter 4, the only two schemes we know of that tackle SC^2R are portrayed.

The first one is entirely based on artificial ant colonies, particularly on the Max-Min Ant System (MMAS) model. Several heuristic rules that indicate the desirability of visiting the next node during the incremental construction of the tour are put forward. Enlightening insights and superior results are obtained when we favour a synergetic view of the system from the heuristic standpoint instead of leaning on a biased stationary transition rule. The algorithm outperforms a state-of-the-art version of Simulated Annealing in efficacy, i.e. it generates solutions of better quality.

Then we strived to attain more remarkable performance gains across a wider array of problem instances and came up with a hybrid approach of genetic algorithms and ant colonies (GA-ACO). Key features of the proposed method are the improved construction of the initial population, the efficient selection of good visiting sequences by means of pheromone-based

crossover, the utilization of general and problem-specific local search operators and the enhanced diversification achieved via generational pre-crossover parent mutation.

Another contribution in this scope lies in the empirical methodology for contrasting GA-ACO with the early ant-based model. The evaluation (described in Chapter 4) is centered around a multilevel regression analysis that uncovers decision rules that dictate which algorithmic components of GA-ACO must be turned on/off to face specific problem instances. Statistically verified results from extensive simulations conducted with over 8,000 data scenarios confirm that GA-ACO excels MMAS at the 1% confidence level.

The extension of the SC^{2}R framework to a multi-robot fleet (termed as *multiple-carrier coverage repair*, MC^{2}R) is another important contribution of this doctoral thesis and becomes the subject of Chapter 5. MC^{2}R gives rise to the formulation of another novel combinatorial optimization problem (*the one-commodity vehicle routing problem with selective pickup and delivery*, 1-VRP-SELPD). While conceptually similar to 1-TSP-SELPD, a solution to 1-VRP-SELPD is more difficult to find given that it has to decide how many sensor replacement trajectories are needed and what nodes each must visit. This fact implied a substantial shift in the design guidelines of the optimization algorithm aimed at tackling it. The two proposed solutions to SC^{2}R in Chapter 4 only navigated across the feasible search space, i.e. invalid solutions were readily discarded. But now 1-VRP-SELPD bears a higher level of infeasibility (due to the construction of multiple routes) yet the time available for the calculations is most likely the same as for the solo-robot variant. This led us to think that infeasible solutions had to be seriously taken into account (i.e. we move across the infeasible space), and a mechanism that gradually drives the system into feasibility is sorely wanted. In that way, the RA-WSN could remain responsive to its real-time operational demands while still yield good-quality sensor relocation trajectories.

A hybrid metaheuristic algorithm is put forward in Chapter 5 to solve MC^{2}R instances. The composite scheme relies on a swarm of artificial fireflies in which each individual follows the exploratory principles featured by Harmony Search (HS). Infeasible route plans are gradually

driven into feasibility under the influence of a weak Pareto dominance relationship. A repair heuristic is finally applied to yield a full-blown solution. As far as we are concerned, this scheme is the first one in literature that tackles the MC^2R problem. Empirical results indicate that promising solutions can be achieved in a limited time span.

The scientific value ascribed to the formulation of 1-VRP-SELPD as a new extension of the Vehicle Routing Problem (VRP), the introduction of MC^2R as a pragmatic and challenging problem in RA-WSNs, its formalization as a unitary 1-VRP-SELPD instance and the proposed hybrid metaheuristic solution were all acknowledged by the IEEE Computational Intelligence Society when it granted the *2011 Walter Karplus Graduate Student Research Grant* to this doctoral candidate in order to carry out these research endeavours.

The SC^2R-related investigation results are reflected in [52], [48] and [49], the two latter being fruits of the academic collaboration with colleagues from the Transportation Research Institute (IMOB), Hasselt Universiteit in Belgium during a visiting scholarship of this doctoral candidate in the summer of 2010. The research outcome for MC^2R can be found in [53].

1.3.3 Identification of the Defective Sensing Units

As stated in Section 1.2.3, the truly sought-after traits of any protocol intending to target system-level fault diagnosis are scalability and efficiency, for the reliability component can be dealt with by arranging the way nodes test each other so as to ensure an unambiguous prediction. In other words, we must devise an approach that can quickly locate the true set of faulty units in a parallel or distributed system regardless of its size and topological structure. Existing solutions are either exact methods (able to solve only small instances) or metaheuristic algorithms applicable to larger problems but whose memory and time demands may be prohibitive in many resource-restricted environments.

Our first attempt came in the form of a binary particle swarm optimization (PSO) scheme, as explained in Chapter 6. The simplicity of this bio-inspired method and its demonstrated skills in overcoming local optima throughout the search space [78] made possible to beat a

previous implementation rooted on artificial immune systems (AIS) when tested on small and medium-size networks. PSO probes a much lower number of candidate solutions to arrive at the global optimum and does it faster than AIS. It uses a binary encoding to represent the particles' position and applies a stochastic rule to the real-valued particle velocity in order to clamp the positions of the swarm members to the binary search space. Furthermore, it rapidly conducts any infeasible encoding into feasibility, thus accelerating the convergence rate.

Despite its good detection performance in the average case, PSO suffers from an erratic worst-case behavior in medium-sized systems and a lack of convergence in large systems, these caused by inefficient guidance from the best particle. Both drawbacks were subdued in Chapter 6 too by fostering a stronger degree of social interaction among the search agents. We relied on a recently formulated algorithm called "Firefly Optimization" (FO) [149], which can be seen as a natural extension of PSO in which each agent (firefly) feels attracted not only to the best individual in the swarm (as in PSO) but to every other firefly "brighter" than itself. The brightness has to do with the quality of the solution found (fitness function). By expanding PSO's social perspective with the FO dynamics and customizing the light absorption coefficient (a built-in FO parameter) to the needs of every firefly in the swarm, a scalable fault diagnosis method was conceived. It was tested in small, medium and large graphs and yielded very encouraging results. FO's low memory requirements and enhanced convergence pace makes it a suitable candidate to be run in resource-constrained settings, such as a cluster-head node in a WSN that is responsible for verifying the integrity of all nodes within its cluster.

Another thesis contribution lies in the risk management framework described in Chapter 7. The envisaged architecture is comprised of several modules but we focused our efforts in the specifications of an *adaptive risk visualization* and a *real-time risk assessment* modules. This is, to the best of our knowledge, the first ever proposed risk framework that utilizes clusters corresponding to the sensor nodes in the risk feature space as a visual depiction mechanism that triggers the human intervention upon the system's risk perception for any sensor, group of sensors or the entire WSN. The application of an evolving clustering approach in the con-

text of risk management to accommodate the knowledge structures (clusters) in the previous data snapshot to the reality pervading the current data snapshot is also unprecedented. This is accomplished by means of a *dynamic cluster formation* process that involves cluster deletion, merge, split and outlier detection. Shadowed sets are adopted as the underlying formalism for representing the clusters given their proved robustness to noise and outliers, high interpretability and fast convergence. Furthermore, the risk quantification for any system unit is provided by the risk assessment module on the basis of a fuzzy and/or shadowed modeling of the risk features and a simple learning process that adapts the features' weights in proportion to their impact on observable failed sensors is put forward as well.

The research findings accomplished in system-level fault diagnosis have been voiced in [46] and [47] whereas our risk management framework for WSNs was published in [45].

1.4 Outline

The remainder of this manuscript has been structured as follows:

Chapter 2 navigates the reader across those technical concepts and previously published research studies that are relevant to the thesis.

Chapter 3 is devoted to the focused coverage formation problem and the proposed distributed algorithm.

Chapter 4 addresses the important problem of coverage repair in RA-WSNs and elaborates on the two nature-inspired strategies that tackle the single-carrier scenario.

Chapter 5 enunciates the multiple-carrier coverage repair problem and puts forth a hybrid method for it.

Chapter 6 deals with the fault detection process in t-diagnosable systems and sheds light on the two swarm intelligence methods that are claimed as contributions.

Chapter 7 unveils the novel risk management framework and its application to a simulated Territorial Security scenario.

Chapter 8 wraps up the thesis and enunciates potential research directions.

CHAPTER 2

LITERATURE REVIEW

The major areas related to the present doctoral thesis are surveyed in this chapter. They have to do with topics like sensor placement and relocation by mobile actuators, combinatorial optimization scenarios (in particular, traveling salesman and vehicle routing problems with pickup and deliveries), fault identification in diagnosable systems, distributed artificial games, nature-inspired metaheuristic optimization approaches, granular computing (in the form of shadowed sets) and evolving fuzzy clustering techniques. Despite their apparent dissimilar nature, these fields can be seamlessly interwoven in the novel fabric of *fault-reactive wireless sensor and robot networks*.

The reader will first encounter a concise yet descriptive taxonomy of WSN protocols, followed by the techniques employed to tackle the core problems that surfaced along this doctoral study. The chapter goes on by briefly discussing the state-of-the-art in each of the aforementioned research areas.

Let the journey begin.

2.1 Technical Background

This section equips the reader with the necessary technical background that is required to fully understand the thesis contributions presented in Chapters 3 - 7.

2.1.1 A Categorization of WSN Algorithms

An algorithm for a WSN can be classified as *centralized*, *distributed* or *localized* depending on the scope of the network knowledge required [86].

In centralized solutions, local information flows from each node in the network to a central unit (e.g. a base station or clusterhead device), which is responsible for building a global view of the system. The role of the sensors is to convey the demanded data to the main controller, possibly through multi-hop paths. The controller is assumed to be resource-rich, therefore being able to carry out entangled operations. Its output could be communicated to a subset (or all) of the network nodes so that further action can be taken. For instance, the two CBCR protocols in Chapter 4 share a centralized nature because the calculation of the optimal robot trajectory happens at the base station. Upon completion, the controller then commands a mobile robot to move around the field and replace damaged sensors with passive ones. Centralized approaches usually incur in large communication overhead when collecting the desired data and propagating the output, which is not good for scalability. Also if the controller goes down, the algorithm renders infeasible. However, they provide accurate results and are still an active trend in WSN research.

Distributed algorithms, on the other hand, allow each network entity to form its own picture of the WSN by interacting with peers. There is no central unit in charge of a global task. Any node might require data from every other node (as in a centralized protocol, but now the algorithm runs distributedly) or from its immediate neighbors (up to k hops). Distributed solutions are more reliable (i.e. no bottleneck or single-point failure) than centralized schemes yet the overhead caused by message passing could still be significant, depending on what problem

they are solving (e.g. construction of a fault-aware spanning tree). There is also a constraint on the resources locally available to the sensor nodes, which limit the extent of the operations that can be delegated to them. Chapter 6 portrays a fault diagnosis framework where the node testing phase is distributed but the node labeling stage is centralized.

Finally, a localized protocol is inherently distributed but needs not global communication nor computation because both are restricted to a particular vicinity of every node. The entailed overhead is low (thus providing for scalability and adaptability) but the results are only reflective of the neighborhood structure. This accuracy-overhead tradeoff is oftentimes ruled in favor of the latter given the physical limitations of the sensory units. As a result, a surge of innovation in the design of localized algorithms is sweeping the literature, including the one presented in Chapter 3.

2.1.2 Bio-inspired Metaheuristic Optimization

Challenging optimization problems of single and multi-objective character demand more aggressive search strategies than those offered by traditional methods (e.g. gradient-based and exact algorithms). While gradient-based schemes may get easily trapped in local minima as they explore an irregular landscape, exact algorithms can only cope in practice with small problem instances. The need for *bio-inspired* (or nature-inspired) methods stems from the above limitations. They are regarded as *metaheuristics* because of the high-level search strategies (heuristics) that define their behavioral patterns [150], which are applicable to a wide range of optimization scenarios with good results in general.

Briefly defined, a bio-inspired metaheuristic algorithm is one that exploits design principles present in biological systems and communities of living animals (including humans) in an orderly fashion so as to improve the convergence of the search process over the solution space [134] [150]. Three main branches of bio-inspired computing are envisioned according to [134]: *evolutionary algorithms* (EAs), *swarm intelligence* (SI) techniques and *bacterial foraging* (BF) optimization. This study makes active use of representative approaches from the

first two categories.

EAs [34] heavily cling to the phenomenon of genetic evolution, in which species survive based on their adaptation to the environment [134]. They reproduce and mutate as a way of guaranteeing their survival and diversification. The family of EAs include genetic algorithms (GAs) [62], genetic programming (GP), evolutionary strategies (ES) and evolutionary programming (EP). They rely on a population of individuals which iteratively evolves over multiple generations after the application of genetic operators.

SI techniques [18] are anchored on animal behavior, particularly on the complex objective achieved by simple, individual agents that interact either directly or indirectly with one another. Ant colonies, bird flocks, fish schools, bat clouds and bee nests are examples of SI systems in which the communication patterns ingrained at the individual level make possible to accomplish global tasks at the population level whose complexity far exceeds that of its constituent agents.

In the rest of this section, we describe in further detail those EAs and SI-based schemes that have been employed as part of this thesis. We will also elaborate on Harmony Search, a recently proposed non-population-based optimization approach inspired by the quest for good pitches undertaken by musicians in real life.

Although evolutionary and swarm intelligence algorithms have been widely employed to optimize operational aspects of a wireless sensor network (e.g. data aggregation, sensor localization, energy efficiency during communications) and boost the behavior of robotic agents (e.g. path planning, assembly line balancing, task allocation, autonomous navigation, etc.), their application to the emergent realm of WSRNs seems to be rather unexplored yet. The present doctoral study intends to spark further research endeavors in this promising area.

Genetic Algorithms (GAs)

Genetic Algorithms are one of the most popular representatives of evolutionary optimization methods. Originally introduced by Holland [73], this metaheuristic approach is rooted on

the principles of Darwinian evolution such as selection of the fittest, reproduction and mutation. A population of *chromosomes*, which represent candidate solutions to the optimization problem, evolves over a number of generations through the application of genetic operators. Each chromosome encodes a candidate solution as a multidimensional vector and may additionally embed some algorithmic parameters. Binary and real-valued encoding schemes are the two most popular ones in literature [62]. The former deals with strings of bits while the latter utilizes either integer or floating-point numbers for each *gene* (vector component). Extracting information from a chromosome leads to diverse decoding methods [151]. The total number of genes in a chromosome could be fixed [154] or variable [23].

Genetic operators aim to ensure that the population is diverse, thus better avoiding premature convergence. They include survival of the fittest, crossover and mutation [62]. At the end of each generation, the best individuals are allowed to pass to the next generation and the poorest are trimmed. Crossover is the exchange of multiple genetic sequences (of the same length) between two parents in the population, hence giving rise to offspring individuals. One or more crossing points are predetermined, often arbitrarily. Mutation affects a single individual sporadically by altering one or more genes as a mechanism to increase diversification. Crossover and mutation occur probabilistically, the former more likely than the latter.

The widespread belief is that the performance of GAs (and of EAs in general) largely rests upon the design of the crossover operator and several techniques (e.g. incest prevention, informed crossover) have been put forward to enhance it [104]. Among them, *swarm-based crossover operators* are earning a sound reputation. In particular, the incorporation of pheromone information from ACO systems (see Sect. 2.1.2) reported significant improvements in the convergence capabilities of GP [144] and GAs [154] [155]. Therefore, such perspective has been considered in our research study as well.

Ant Colony Optimization (ACO)

Ant colony optimization is a stochastic search technique [39] to find shortest paths in discrete (graph-representable) optimization problems. It owes its inspiration to the foraging behavior of natural ants, whose high degree of coordination makes possible to attain good sub-optimal solutions (i.e., short paths from the nest to the food source) at the colony level. Communication is performed indirectly, i.e., mediated by the environment (*stigmergy*) through the deposit of a chemical substance named pheromone. The higher the intensity of the pheromone deposited on a path, the more likely it is to be selected by future ants as part of their journey.

Ants are initialized in different graph nodes, either at random or following a specific rule. Their purpose is to concurrently and incrementally construct tours (i.e., candidate solutions). The next node in the tour is selected in a stochastic fashion according to a probabilistic transition rule from the current node to every other node in each ant's neighborhood (set of nodes to which the ant can move from the current node). Such rule is posed as a function of the pheromone trails already present in the graph and the problem-dependent heuristic desirability of going from one node to another. In the Ant System (AS), the first ACO algorithm [38], the transition rule takes the form:

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}]^{\beta}} \,, \tag{2.1}$$

where p_{ij}^{k} denotes the probability of the k-th ant selecting j from i as the next node in the tour among all candidates in its neighborhood N_i^k, $\tau_{ij}(t)$ is the amount of pheromone deposited on e_{ij} at iteration t, η_{ij} is the heuristic desirability of traversing e_{ij} whereas α and β are two sensitivity coefficients associated with the pheromone (collective knowledge) and heuristic (problem-specific knowledge) components of the ACO model, respectively.

The pheromone trails are dynamically altered by the ants during the algorithm's execution, in contrast to the heuristic information that remains usually unchanged. An ACO iteration comprises ant initialization, solution construction and pheromone update. Multiple iterations

take place until some stop condition is met. Defining when, how and by whom the pheromone trails are updated is a pivotal issue for any ACO implementation. For instance, in the elitist version of AS [109], the ant yielding the best solution (i.e., shortest tour) at the end of every iteration is the only one entitled to perform the pheromone update as follows:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij} \,, \tag{2.2}$$

where $\tau_{ij}(t+1)$ is the amount of pheromone on e_{ij} at the next iteration, $0 \leq \rho \leq 1$ is the pheromone persistence rate (implying that some amount will evaporate as a mechanism to prevent stagnation in local optima) and $\Delta\tau_{ij}$ is the change in pheromone concentration on e_{ij}.

The termination of ACO algorithms is often envisioned in terms of a maximum number of iterations carried out, although reaching a threshold on runtime, number of generated tours or solution quality may also become a valid stop criterion.

Max-Min Ant System (MMAS)

Early ACO versions exhibit a degraded performance for problems of large dimensions owing to the stagnation phenomenon which arises due to the long-term accumulation of pheromone on the best edges (especially in elitist approaches, i.e., those that reward the best solutions alone). This undesirable effect strongly biases the ants' selection and narrows their overall exploration capabilities. Premature convergence to suboptimal solutions has been dealt with the Max-Min Ant System (MMAS) introduced by Stützle and Hoos [131]. MMAS can be regarded as a direct successor of AS (actually, both use the same transition rule (2.1)) yet it introduces innovative modifications to the AS machinery.

MMAS is an elitist approach, i.e., it highly exploits the most promising (either iteration-best or global-best) tour discovered, whose edges receive the highest pheromone concentrations. However, it strongly encourages the exploration of the entire space by capping the amount of pheromone on each edge such that $\tau_{ij} \in [\tau_{min}; \tau_{max}]$. The obvious effect is that there will also be a cap on the probability of an edge being selected as next node. Pheromone trails are initialized to τ_{max} which increases the exploration of tours at the beginning of the search.

Furthermore, the trails are reinitialized each time the system undergoes lack of improvement of the best tour generated after a certain number of consecutive iterations. A remarkable MMAS feature is the dynamic character of the pheromone bounds, which change every time a global best solution is encountered.

MMAS behavior was thoroughly examined for several TSP instances [131] and it was proved to excel competitive ACO algorithms, including Ant Colony System (ACS).

Particle Swarm Optimization (PSO)

A landmark representative of swarm intelligence approaches [150] is Particle Swarm Optimization (PSO), which exploits the social behavior of some animal communities like bird flocks or fish schools in the pursuit of specific tasks, e.g. birds flocking to a food source. The inherently collaborative nature of these biological systems led to the design of an efficient optimization scheme for exploring non-linear, non-differentiable, multimodal, and multidimensional real-valued search spaces. Originally introduced by Kennedy and Eberhart in 1995 [78], PSO has earned a well-deserved renown among population-based metaheuristics for its intuitive description, algorithmic simplicity and proven competence to reach the global optimum in a wide array of challenging optimization landscapes. Hence, it has become one of the most active research subjects in bio-inspired optimization models.

In PSO, a swarm S of search agents (particles) concurrently moves throughout a continuous landscape in its quest for the optimal solution of a numerical function. The swarm is composed of $n = |S|$ particles, each representing a candidate solution to the optimization problem. The i-th particle is defined by a position vector $\vec{X}_i = (X_{i1}, X_{i2}, \ldots, X_{id})$ (where d is the dimensionality of the search space) and a velocity vector $\vec{V}_i = (V_{i1}, V_{i2}, \ldots, V_{id})$. The position of every agent feeds a problem-dependent fitness function $f : \mathfrak{R}^d \rightarrow \mathfrak{R}$ and each particle remembers its best-ever-found position $\vec{P}_i = (P_{i1}, P_{i2}, \ldots, P_{id})$. This is the *cognitive* component of PSO, often referred to as the particle's "*local best*".

Additionally, a neighborhood is defined for every particle as the subset of individuals

which it is able to communicate with. The first PSO model used an Euclidean neighborhood by measuring the actual distance among particles (as an imitation of the biological models) to determine the nearest individuals. This formulation was indeed computationally intensive and became eventually replaced with topological neighborhoods, which do not regard the particles' physical vicinity. Among them, the so-called "*global-best*" topology allows full communication among all swarm individuals. A good analysis of "local-best" versus "global-best" neighborhood definitions can be found in [20]. The kind of neighborhood adopted for a particular PSO implementation determines its "*social*" component.

Particles fly along the search space attracted by their best individual solution $\vec{P}_i$ ("local best") and the best solution $\vec{P}_g$ found by any particle in their neighborhood ("neighborhood best"). The entire swarm is updated at each iteration by modifying the particles' velocity and position vectors along every dimension as per the following rules:

$$V_{ij} = \chi \left(V_{ij} + c_1 \cdot \epsilon_1 \cdot (P_{ij} - X_{ij}) + c_2 \cdot \epsilon_2 \cdot (P_{gj} - X_{ij}) \right) \tag{2.3}$$

$$X_{ij} = X_{ij} + V_{ij} \tag{2.4}$$

where c_1, c_2 are called "acceleration constants", ϵ_1, $\epsilon_2 \sim$ U(0,1) are independent random numbers uniquely generated every time for each dimension $j \in \{1 \dots d\}$, χ is the "constriction factor" meant to balance the global and local exploration of the algorithm. This parameter plays a fundamental role in PSO achieving good convergence capabilities and can be computed as shown in (2.5). Recent implementations set $c_1 = c_2 = 2.05$ to obtain $\varphi > 4$ and thus guarantee convergence, for otherwise the swarm would "spiral" toward and around the best solution found in the search space [20].

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \varphi = c_1 + c_2 \tag{2.5}$$

The algorithm terminates usually after a certain number of iterations. Notice that PSO does not generate new individuals unlike evolutionary approaches but dynamically modifies

the location of its swarm members, which alleviates the time and space complexity when dealing with overly complex systems.

Binary PSO

Though at first conceived for numerical optimization problems, PSO has undergone many developments since its inception that make it now fit for targeting combinatorial scenarios. The first discrete PSO was proposed by its authors in [79] shortly after its birth.

In discrete PSO, each particle is encoded as a binary vector of length d and the position update equation in (2.4) is redefined as a probabilistic rule based on a sigmoidal transformation applied to the real-valued particle velocity.

$$X_{ij} = \begin{cases} 1, & \text{if rand( )} \leq \dfrac{1}{1 + \exp(-V_{ij})} \\ 0, & \text{otherwise} \end{cases} \tag{2.6}$$

where rand( ) $\sim$ U(0, 1). Expression (2.6) forces the particle's components to be binary values while still allows the swarm members to benefit from the cognitive and social components of the original PSO formulation for the velocity calculation in a continuous search space. The simplicity behind this idea is one of the major appeals of the algorithm, which in turn has become the focus of more intensive research for speeding up convergence.

Firefly Optimization (FO)

This metaheuristic was proposed by X. S. Yang in 2009 [149] inspired by the phenomenon of *bioluminescent communication* featured by firefly insects in tropical countries. It has been observed that most fireflies produce short and rhythmic flashes, which serve to attract either potential prey or other fireflies for mating purposes. Since the light intensity (brightness) is inversely proportional to the square of the distance from the light source (firefly) and a certain amount of light is absorbed by the air, fireflies are only visible up to a limited distance.

Bearing the above ideas in mind, FO was enunciated as a population-based optimization method relying on a swarm S of size $n = |S|$, in which each candidate solution (firefly) has an

associated position vector $\vec{X}_i = (X_{i1}, \ldots, X_{id})$, $i = 1, \ldots, n$ in the real-valued, d-dimensional continuous search space. The *brightness* of any firefly $I_0(\vec{X}_i)$ is directly/inversely proportional to its fitness value $f(\vec{X}_i)$ for the maximization/minimization case. The degree to which $\vec{X}_i$ is attracted towards another firefly $\vec{X}_j$ is modeled after the distance-dependent *attractiveness attenuation function* $\beta(\vec{X}_i, \vec{X}_j)$ as follows:

$$\beta(\vec{X}_i, \vec{X}_j) = \beta_0 \cdot \exp(-\gamma \cdot r_{ij}^2) \tag{2.7}$$

where β_0, γ are parameters called "initial attractiveness" and "light absorption coefficient", respectively (both usually set to fixed values) and $r_{ij} = \text{dist}(\vec{X}_i, \vec{X}_j)$ is a distance measure (e.g. Euclidean) between $\vec{X}_i$ and $\vec{X}_j$.

FO can be viewed as a generalization of PSO because any swarm member may consider multiple "informers" in its neighborhood for updating the position vector, whereas the standard PSO only regards a single informer: the best individual in a particle's neighborhood.

More precisely, each firefly $\vec{X}_i$ is attracted along any problem dimension $k = 1, \ldots, d$ towards every other firefly $\vec{X}_j$ brighter than itself as shown in (2.8):

$$X_{ik} = X_{ik} + \beta(\vec{X}_i, \vec{X}_j) \cdot (X_{jk} - X_{ik}) + \alpha \cdot \left(\text{rand}(\,) - \frac{1}{2}\right) \tag{2.8}$$

where α is a randomization factor (built-in parameter) which controls the extent of the stochastic perturbation endured by X_{ik} and rand( ) $\sim U(0, 1)$. Notice that the above expression is computed only when $I_0(\vec{X}_j) > I_0(\vec{X}_i)$. In the canonical FO formulation, a "global-best" neighborhood topology is adopted.

The above movement law does not apply to the best firefly $\vec{X}_{i^*} \in S$, as no one is "brighter" than itself. In such case, the "social" term in (2.8) is dropped, leaving $\vec{X}_{i^*}$ the freedom to perform a controlled random walk across the search space as shown in (2.9).

$$X_{i^*k} = X_{i^*k} + \alpha \cdot \left(\text{rand}(\,) - \frac{1}{2}\right) \tag{2.9}$$

Discrete Firefly Algorithms

The only discrete firefly implementation known at the time of this writing is that in [126]. It was proposed for makespan minimization in permutation flow shop scheduling problems. Building upon FO's verified success in dealing with entangled combinatorial and numerical optimization landscapes, two discrete FO algorithms are presented in this doctoral study.

Chapter 6 elaborates on a binary version (i.e. each swarm member is encoded as a multidimensional binary vector) for detection of faulty units in diagnosable systems. It keeps a close resemblance to that in [126]. After receiving guidance from every informer firefly $\vec{X}_j$ as in (2.8), the real-valued position vector $\vec{X}_i$ is transformed into a binary one by means of a probabilistic rule whose expression is given in (6.6).

Chapter 5 introduces a discrete FO formulation for the multiple-carrier coverage repair problem. Each firefly denotes a set of disjoint routes (i.e. route plan) followed by a team of mobile robots to substitute damaged sensors in the WSN field with spare ones. A cyclic integer-valued representation is utilized for such purposes. Moreover, FO is mingled with the algorithmic simplicity and enhanced exploratory principles behind Harmony Search to provide high-quality solutions to difficult MC^2R problem instances.

Harmony Search (HS)

HS is a simple yet powerful music-inspired metaheuristic scheme [60], since the improvisation process of any musician in his pursuit of harmony is analogous to the search process in the optimization realm.

A candidate solution (harmony) $\vec{X}_i$ is a collection of pitches $(X_{i1}, \ldots, X_{iN})$. To improvise a new pitch X_{ij}, the musician has three options: (1) to draw a good pitch from the Harmony Memory (HM), an elitist data structure containing HMS high-quality harmonies; (2) to slightly adjust the previously chosen pitch or (3) to come up with a brand new pitch.

Good past pitches are drawn from the HM with a probability HMAR (HM acceptance rate) and subsequently twisted with probability PAR (pitch adjustment rate). The third choice

is done entirely at random. The pitch improvisation is repeated $\forall j,\ j = 1..N$ until an entire harmony $\vec{X_i}$ is completed. It replaces the worst harmony in HM if $\vec{X_i}$ is of superior quality.

Because of the generic nature of options (1) – (3), HS could target both discrete and numerical optimization domains. Notice that option (1) is strongly related to the exploitation capability of the algorithm, as it intends to reuse good solution components (pitches) stored in the HM. On the other hand, HS' exploratory power largely rests on options (2) and (3), which add perturbation to a HM-drawn pitch or generate a new one from scratch, respectively.

Given its algorithmic simplicity and fast convergence rate to the global optimum across diverse optimization landscapes, HS has been hybridized with several metaheuristic approaches, including genetic algorithms [106], simulated annealing [76], tabu search [82] and differential evolution [123]. In Chapter 5, HS is amalgamated for the first time with a swarm of artificial fireflies in order to solve challenging instances of the multiple-carrier coverage repair problem.

2.1.3 Shadowed Sets

Shadowed sets are one of the most recent manifestations of Granular Computing [116]. They are algorithmically induced by fuzzy sets [117] and were created to counter the uncertainty arising from the assignment of quantitative membership degrees $\mu_X(x) \in [0, 1]$ to any element x of the concept X. A shadowed set performs two cuts over the membership grade space of a fuzzy set X, thus splitting it into three disjoint regions: the *core* CORE(X), the *exclusion* XCL(X) and the *shadowed* SDW(X). Each element $x \in X$ belongs to exactly one region, i.e. $\text{CORE}(X) = \{x \mid \mu_X(x) \geq 1 - \lambda\}$, $\text{XCL}(X) = \{x \mid \mu_X(x) \leq \lambda\}$ and $\text{SDW}(X) = \{x \mid \lambda < \mu_X(x) < 1 - \lambda\}$, where $\lambda \in (0, 0.5)$ is a threshold value. The creation of a shadowed set is depicted in Fig. 2.1.

The membership degrees of the core elements are elevated to 1, those of the exclusion members are lowered to 0 and those of the shadowed elements remain as in the original fuzzy set X. These three zones are functionally equivalent to the positive, boundary and negative regions in rough sets [115]. They are, therefore, a theoretical bridge between the fuzzy and

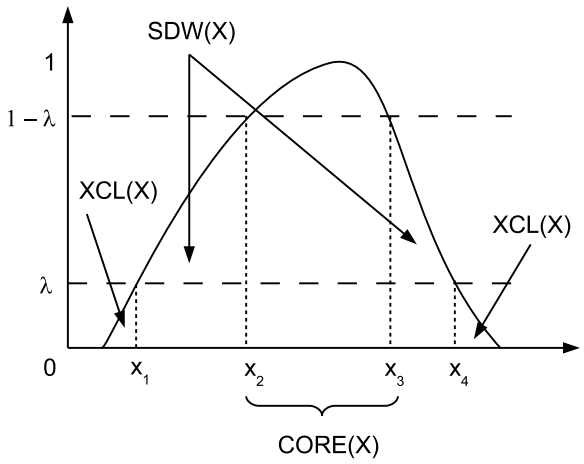


Figure 2.1: Development of a shadowed set associated with concept X. The regions induced by threshold λ are: $\text{CORE}(X) = \{x_2 \leq x \leq x_3\}$, $\text{XCL}(X) = \{x \leq x_1\} \cup \{x \geq x_4\}$, $\text{SDW}(X) = \{x_1 < x < x_2\} \cup \{x_3 < x < x_4\}$

rough set formalisms.

The optimal value of λ is sought after the principle of *uncertainty relocation* proposed by Pedrycz [115], i.e. the reduction and elevation of membership in the exclusion and core zones must be compensated with the uncertainty remaining in the shadowed region. The optimal value of λ is the one that minimizes (2.10), where $\mathbf{x}_k$ is a data pattern in a finite-size collection ($k = 1..N$) and μ_X^{max} is the maximum membership grade of any data pattern to the concept X.

$$O(\lambda) = \left| \sum_{\mathbf{x}_k \mid \mu_X(\mathbf{x}_k) \leq \lambda} \mu_X(\mathbf{x}_k) + \sum_{\mathbf{x}_k \mid \mu_X(\mathbf{x}_k) \geq \mu_X^{max} - \lambda} (\mu_X^{max} - \mu_X(\mathbf{x}_k)) - \text{card}\left(\mathbf{x}_k \mid \lambda < \mu_X(\mathbf{x}_k) < \mu_X^{max} - \lambda\right) \right| \tag{2.10}$$

Shadowed sets are more interpretable granules than fuzzy sets. They bear the qualitative strength of rough sets with the added value of numerical discernibility in the shadowed regions provided by fuzzy sets. For these reasons, they have been recently adopted to characterize fuzzy clusters [105] as well as rough and rough-fuzzy clusters [156] in "offline" mode. They have exhibited a fast convergence rate to the global optimum during the optimization-driven clustering machinery and high robustness versus noise and outliers.

In Chapter 7, shadowed sets are employed in the visualization module of the proposed risk-aware management framework to group sensor nodes across the risk feature space in an online, evolving fashion and also in the assessment module to provide local perceptions of the risk features under consideration.

We will briefly describe the recently proposed Shadowed C-Means (SCM) clustering algorithm [105]. It is an offline clustering machinery that becomes the major building block for the risk visualization module in Sect. 7.2.

Shadowed C-Means

SCM is a partitive clustering algorithm in which each cluster is represented as a shadowed set. It needs the desired number of clusters c as input and returns a set of cluster prototypes $V_1, V_2, \ldots, V_c$, a collection of shadowed thresholds $\lambda_1, \lambda_2, \ldots, \lambda_c$ and a partition matrix $U = (u_{ik})$ holding the membership grade of each data pattern X_k to every prototype V_i, where $k = 1..N, i = 1..c$ and N is the number of patterns.

The degrees of membership in SCM are computed as in (2.11), where m is the fuzzifier parameter and d_{ik} is any distance measure from X_k to V_i.

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{d_{ik}}{d_{jk}}\right)^{2/(m-1)}} \tag{2.11}$$

During the prototype formation process, SCM weighs the patterns based on the region they belong to, as in (2.12).

$$V_i = \frac{\sum\limits_{X_k \in CORE(X)} X_k + \sum\limits_{X_k \in SDW(X)} u_{ik}^{m} X_k + \sum\limits_{X_k \in XCL(X)} u_{ik}^{m^m} X_k}{|CORE(X)| + \sum\limits_{X_k \in SDW(X)} u_{ik}^{m} + \sum\limits_{X_k \in XCL(X)} u_{ik}^{m^m}} \tag{2.12}$$

This implies that SCM is strongly biased towards core patterns (as they get the maximum weight 1) and is robust to noise and outliers because of the low importance granted to exclusion patterns. SCM is a nearly parameterless method (c is required by all partitive methods and

m by all fuzzy partitive algorithms) with high interpretability, fast convergence and a more realistic modeling of the available data. For such reasons, it was chosen as the main building block of the evolving clustering architecture that is described in Sect. 7.2.

2.2 Relevant Work

This section goes over the published works that are pertinent to our research efforts.

2.2.1 Sensor Placement by Mobile Actuators

This is one of the four subproblems identified in [91] as falling under the umbrella of *movement-assisted sensor placement*. It has to do with mobile agents setting up the initial, connected layout of a WSN with certain coverage goal in mind. They carry a number of sensors and drop them at desired coordinates (e.g. vertices of a geographic graph) as they travel all over the field.

Movement-assisted sensor placement was studied in the literature in a very limited context. Most of the previous works addressed mobile sensor self-deployment [29,92]. Only a few shed light on sensor deployment by mobile actuators and we will review them at short length.

Batalin and Sukhatme [14] proposed a randomized approach named "Least Recently Visited" (LRV). Sensors store a weight for each direction that a robot can travel in. The weight represents the number of times that a robot has locally visited in/from that direction. A sensor recommends the direction with the lowest weight (i.e. the LRV direction) for a robot to travel. Among all recommendations, the robot randomly chooses one with lowest weight and drops a sensor if current location is not covered. LRV requires many unnecessary movements to fully explore a given environment and construct a full coverage. These extra movements lead to an extremely large number of messages. Furthermore, it is unclear under what conditions LRV terminates. Besides, LRV is a single-robot protocol. Multi-robot extension was mentioned but for a different purpose.

Chang et al. [25, 26] came up with a deterministic snake-like algorithm. The robot gradually moves along a pre-computed graph, each step to an adjacent empty vertex in a predefined order of preference, and drops a sensor after each step. Should the preference order be changed, the robot trajectory will exhibit a different shape such as 'S' shape [25] or spiral shape [26]. This algorithm is very likely to get stuck at dead ends (i.e. where no adjacent empty vertices exist) due to obstacles or early dropped sensors. Dead-end recovery was however not discussed; the algorithm does not guarantee full coverage as a result. It does not support multiple robots either. Shiu [128] later ascribed the lack of coverage guarantee of this approach solely to the presence of concave regions and suggested to treat each concave region as a separate environment. Yet the obstruction of obstacles and sensors to robot movement is overlooked and coverage guarantee is still not accomplished.

Fletcher et al. [56] devised a back-tracking-based sensor placement scheme. The algorithm adopts a snake-like robot trajectory similar to [25] and incorporates a novel back-tracking technique for dead-end recovery, which was implemented via locally stored back pointers on sensor nodes and became further augmented by a shortcut method for boosting efficiency. The algorithm guarantees full coverage if there are enough sensors to cover the area.

Corke et al. [32] authored a connectivity repair procedure using a single mobile robot that deploys sensors. Each sensor maintains a token (initially equal to its unique ID), floods it over the network and updates it to the largest token received. As a result, disconnected regions will have different token values. The robot sweeps the sensory field and collects unique tokens. If more than one is gathered, it knows that the network is not connected and extra sensors are to be deployed. It then estimates the location of the gap by reckoning the coordinates of the border nodes. Another method is to let sensors generate a potential field that pushes the robot to locations where sensor placements are needed. Both methods are centralized at the robot level, rely on frequent token update (communication overhead) and require the robot to have unbounded memory to store information on all nodes for the ensuing computations.

Mei et al. [101] presented a cluster-based algorithm. The idea is to appoint one robot as

the central controller, which will maintain up-to-date information about the location of other robots within the environment. Whenever a sensor failure is reported to the central agent, it notifies the robot nearest to the failure, which then goes and drops a sensor there. A cluster-based distributed implementation of the centralized version was introduced. This scheme depends on frequent network-/cluster- wide flooding and thus has huge communication burden.

In [129], the authors worry about the fact that a sensor might have been shut down due to malicious attack, in which case its replacement must wait until the threat has been suppressed. An intrusion detection protocol for a specific type of denial of service (DoS) attack runs at the base station, which informs the robot nearest to the failure whether it may proceed to unload a functional sensor at that spot or not.

The above works are all meant to target traditional area coverage, thus not applicable to the F-coverage problem. They require robots to carry all sensors at once, which is clearly an impractical requirement as each sensor has a non-negligible physical dimension and each robot possesses a limited cargo capacity. Furthermore, the only algorithms we know of that address F-coverage formation [87–90, 96] are concerned with MSN and heavily exploit locomotion power to achieve the desired end. The protocol in Chapter 3 uses a robotic fleet with bounded cargo capacity rather than mobile sensors and leans on communication mechanisms instead of mobility (which is much more expensive) to construct the final sensor layout.

2.2.2 Sensor Relocation by Mobile Actuators

Sensor relocation deals with node failures, typically in post-deployment WSN scenarios, i.e., how to replace emerging failed sensors with redundant ones through nodal geographic migration, without topological change [91]. It is another branch of movement-assisted sensor placement [91], sometimes termed “coverage maintenance” [94] or “sensor replacement” [101].

Although several such protocols have been developed for MSN [85, 93, 94, 143, 145] and hybrid networks (i.e. those composed of static and mobile nodes) [84, 142], surprisingly only

one study [57], to the best of our knowledge, is concerned with robot-based sensor relocation.

Fletcher et al present in [57] the first algorithm that does not force robots to hold all sensors at once, but allows them to pick up redundant units as they wander all over the field following an arbitrary trajectory. Borrowing ideas from [14], sensors help robots locate the damaged as well as the redundant nodes. A particular implementation in a grid topology is put forward. In general, both versions have low message overhead and minimize the repair latency yet assume that robots are unconstrained in terms of mobility energy and can thus travel endlessly, which is not true in many real-world scenarios.

Compared to [57], the CBCR model in Chapter 4 entails bigger power expenditures in data reporting to the base station (through multi-hop sensor paths), but makes sure robots seat at the base station where they can replenish their batteries and hence perform long-term sensor replacement. The distinctive features of the novel CBCR framework are outlined in Sect. 4.1.

2.2.3 The Antecedents of 1-TSP-SELPD and 1-VRP-SELPD

The traveling salesman problem (TSP) [124] is a landmark *NP*-complete combinatorial optimization problem aimed at finding the minimal-cost Hamiltonian cycle in a given set of cities. TSP has many applications in logistics, planning, chip manufacturing, etc. [124] and its basic formulation has been generalized to face more realistic and complex scenarios.

Two of such extensions that strongly relate to our research are the *Pickup-and-Delivery TSP* (PDTSP) and the *Prize-Collecting TSP* (PC-TSP). The remainder of this section elaborates on both versions.

The one-commodity PDTSP (1-PDTSP) [114] was introduced in [69] as a TSP departure in which all nodes are either delivery customers or pickup customers. A certain amount of a unique commodity is demanded by the former and provided by the latter. Furthermore, the commodity gathered at a pickup customer can be used to meet the need of a delivery customer. A comprehensive survey on pickup and delivery problems between customer locations is given in [113]. Anily and Bramel [4] studied the 1-PDTSP unitary case in the context of in-

ventory repositioning whereas Chalasani and Motwani [24] referred to it as "*Q-delivery TSP"*. Hernández-Pérez et al proposed a branch-and-cut algorithm for handling small 1-PDTSP instances [71], which was subsequently enriched with local search and incomplete optimization [72]. This latter approach was also outperformed by a hybrid algorithm described in [70]. Yet the best known 1-PDTSP results were reported in [154] with the fusion of evolutionary and swarm intelligence methods.

The PC-TSP is a particular instance of a more general TSP taxonomy labeled *"TSP with profits"* in [54], where the chief assumptions are that NOT all nodes have to be visited and that a certain profit is associated with every graph node. In PC-TSP, the objective is to minimize travel costs while the collected profit is not smaller than a predefined value. The problem was originally enunciated by Balas [11], who adds penalty terms for the unvisited nodes. Two exact algorithms are found in literature. Fischetti and Toth [55] pose the problem as an integer linear programming (ILP) model and propose a branch-and-bound method to find optimal solutions. Bérubé et al [15] develop a branch-and-cut algorithm to solve the undirected PC-TSP. Heuristic algorithms are described in [9] and [36]. Metaheuristic solution schemes are presented in [97] and [28] relying on tabu and clustering search, respectively. More recently, Ausiello et al [8] studied the online version of the PC-TSP, in which requests arrive over time and a server has to decide which requests to serve and in what order, without yet knowing the whole sequence of requests. Valle et al [140] define a selective vehicle routing problem for path planning of multiple mobile sinks in WSNs.

The problem described in Sect. 4.1 does not entirely match any of the previous ones. Rather, it stands as a new combinatorial optimization problem, coined as *"the one-commodity traveling salesman problem with selective pickup and delivery"* (1-TSP-SELPD) [48] and sits right at the junction between PDTSP and PC-TSP. The distinctive trait of 1-TSP-SELPD is the assumption that all delivery customers must be readily served but we visit only those pickup locations which are deemed profitable in terms of total travel cost. This is equivalent to say, from the PC-TSP standpoint, that a delivery customer bears a higher profit than a pickup node.

In an analogous manner to 1-TSP-SELPD, the MC^{2}R coverage repair scenario depicted in Sect. 5.1 gives rise to a new kind of combinatorial optimization problem, baptized as *"the one-commodity vehicle routing problem with selective pickup and delivery"* (1-VRP-SELPD). It is a generalization of the well-studied Vehicle Routing Problem (VRP) [63].

Close to 1-VRP-SELPD in literature are the pickup-and-delivery VRP (PDVRP) [40], the Team Orienteering Problem (TOP) [27] [6], its capacity-aware version (CTOP) [5] and the Capacitated Profitable Tour Problem (CPTP) [5].

Like in PDVRP [114], delivery and pickup customers in 1-VRP-SELPD are unpaired and the latter may be used to meet the demand of the former. Yet in PDVRP all graph vertices are to be visited whereas a 1-VRP-SELPD solution includes all delivery nodes plus any subset of pickups that can meet the total demand in the field.

TOP, CTOP and CPTP are similar to 1-VRP-SELPD in that they may leave some graph nodes out of the set of vehicle routes. These problems are grouped under the label "*VRP with profits*" because a reward/profit is collected after visiting each node. TOP and CTOP aim at maximizing the total profit without violating each vehicle's maximum time constraint whereas CPTP strives to optimize the difference between maximum collected profit and total cost. However, these three problems are not concerned with the demand satisfaction of delivery nodes by using the amount of commodity provided by the pickup nodes. Moreover, 1-VRP-SELPD does not restrict the travel time/distance of the vehicles.

Chapter 5 models the emerging MC^{2}R problem after a special 1-VRP-SELPD instance in which delivery customers correspond to damaged sensors, pickup customers are analogous to passive sensors, vehicles are represented by mobile robots, pickups outnumber deliveries and any demand/supply of the commodity (sensors) is exactly one unit.

2.2.4 Fault Identification in t-Diagnosable Systems

Convergence speed in fault detection under t-diagnosable systems is a highly sought-after goal. All approaches described in this section share the same aim: to locate the set of faulty

nodes given the input syndrome in the least amount of time and with high reliability.

A large stream of research works [2, 138, 147, 148] aims at designing efficient algorithms based on the exploitation of structural properties of the testing graph. Hypercubes, crossed cubes, extended stars and twisted cubes, just to name a few, are among the preferred topologies because they simplify the conceptual design of the proposed solution approaches.

We stick to a more flexible methodology in which no restrictions on the topological features of the system are laid, other than those to guarantee a reliable (i.e. deterministic) result. In this avenue, system-level fault diagnosis is posed as an optimization problem under a particular test model and an arbitrary structure of the testing graph, as explained in Sect. 6.2. Exact, heuristic and metaheuristic methods are needed to explore the discrete search space in polynomial time. The complexity of such approaches is generally higher compared to those leaning on a simplified topology yet their applicability is not confined to a particular setting and still yield satisfactory results.

For example, authors in [77] disclose an exact algorithm of the branch-and-bound type whose time complexity is $O(n^3)$, where n is the system size (number of units). Sullivan [132] introduced a backtracking-based enhancement to [77] which caused the time complexity to drop to $O(t^3 + |E|)$, where t is the number of tolerated faults and $|E|$ the total number of tests carried out in the system. This bound becomes more relevant as fewer units are found defective among a large group of nodes.

Evolutionary methods were applied to system-level fault diagnosis by Elhadef et al [42,43, 146], in particular GA and AIS. Both methods lack an adequate population diversity given the biased exploration brought about by an adaptive mutation operator (openly criticized in [146] but finally adopted). As a result of that, the convergence is slowed down and the worst-case identification of faulty nodes gets severely hindered in large systems composed of hundreds or thousands of units. Memory requirements are another important concern, given the enormous number of individuals generated by the algorithms over time in an attempt to find the optimal solution. Such demeanor is made even worst in the AIS model [146], where the number

of elitist (fittest) antibodies and the number of clones per elite antibody are both equal to the population size. These drawbacks have been eliminated in the algorithms described in Chapter 6, as the number of individuals is never altered and an unbiased exploration takes place.

An ACO-inspired scheme was introduced in [44] and tested with the comparison model. The chief idea is to have every ant construct a full tour of the graph representing the system units and label each node as faulty/fault-free as it goes by. Unlike widely recognized ACO models, the authors use neither pheromone nor heuristic values to perform internodal transitions but to guess the node's status. The disadvantage here lies in the redundant storage of dual sets of pheromone trails and heuristic information per node for each ant journey, which aggravates the memory consumption issue for fairly large systems. Furthermore, it is not clear what is the heuristic value assigned to a node by a "flying ant", as no link between the current and "leaping-to" nodes might exist. Another downside of this meta-heuristic implementation is the handling of infeasible solutions, which are discarded by default, thus wasting valuable information gathered during the ant's tour along the graph. Our proposed approaches require less information at the search agent level (particle/firefly), gracefully turn infeasible solutions into feasible ones and are theoretically simpler than their ant-based fellow.

2.2.5 The Goore Game

The Goore Game (GG) [136] was introduced in 1973 by M. L. Tsetlin as a distributed artificial game with some peculiar properties. The description in [111] reads as follows:

> "Imagine a large room containing N cubicles and a raised platform. One person (voter) sits in each cubicle and a referee stands on the platform. The referee conducts a series of voting rounds as follows. On each round, the voters vote 'Yes' or 'No' (the issue is unimportant) simultaneously and independently (they do not see each other) and the referee counts the fraction λ of 'Yes' votes. The referee has a unimodal performance criterion $G(\lambda)$, which is optimized when the fraction of 'Yes' votes is exactly λ^*. The current voting round ends with the referee awarding

> a dollar with probability $G(\lambda)$ and assessing a dollar with probability $1 - G(\lambda)$ to every voter independently. On the basis of their individual gains and losses, the voters then decide, again independently, how to cast their votes on the next round."

As stated in [111], the Goore Game is a non-trivial non-zero-sum game which, unlike traditional AI games such as Chess, Checkers, etc., is completely distributed. This means that the players (voters) are unaware of the game configuration like number of players, payoff function, etc. Since the players get rewarded or penalized based on the decisions they make, Learning Automata [107] are often used to represent the interactions between a player and the referee. The stochastic function used to reward or penalize the players can be completely arbitrary as long as it is unimodal. Yet the most exciting feature about GG is that should each voter update its action based on either a Tsetlin automaton with large memory or an absolutely expedient algorithm, then the entire group will asymptotically optimize the referee's performance criterion.

Researchers have already taken advantage of the GG's non-cooperative and self-organizing nature in order to provide Quality of Service (QoS) support in WSNs [75] and timely coordination for groups of mobile robots [137]. More recently, authors in [21] claim to have designed for the first time a search algorithm using non-cooperative players (GG), as opposed to the traditional collaborative and competitive strategies (e.g. metaheuristic algorithms). Their application is related to the identification of randomly located objects scattered over large areas.

Oommen et al have removed a few years ago one of the major barriers for the application of GG to manifold scenarios. This limitation had to do with the fact that the accuracy of the asymptotic solution obtained in a distributed fashion is intricately in connection with the number of players participating in the game. In other words, an arbitrary accuracy can only be achieved if the game has an infinite number of players. By using no more than three stochastic learning machines and recursively pruning the solution space to ensure that the retained domain contains the solution to the GG as likely as possible, authors in [110] [111] have attained

an unbounded accuracy for GG, thus presenting the first practical GG implementation and paving the way for its application to a variety of scientific areas.

In light of the aforementioned research findings, one could envision a GG-based implementation of the distributed fault detection framework outlined in Sect. 2.2.4. That is, each unit in the WSN under consideration is regarded as a voter that chooses to adopt a binary labeling about itself (faulty/fault-free) and that receives feedback (in the form of a reward or a penalty) about its belief directly from the supervisory unit. Each node then adjusts its individual labeling over time (independent from what other nodes believe) and informs the supervisory unit accordingly.

Nevertheless, a more thorough analysis of the problem at hand will reveal that the cooperative, metaheuristic-based approach undertaken in Chapter 6 is still preferable to the decentralized GG scheme for the following two reasons:

(1) **Limited execution time**: The convergence time of GG to the system-wide solution (global optimum of the unimodal referee's performance criterion) is usually long and we cannot afford such delay in order to identify the set of faulty units in the system. The two swarm intelligence algorithms described in Chapter 6 manage to find the optimal solution very quickly.

(2) **Negligible communication overhead**: While the GG-based approach requires regular communication expenses (i.e., update from each sensor to the supervisory unit at every iteration and the ensuing broadcast from the supervisory unit to all sensors in order to convey the feedback), the nodes in the cooperative scheme submit their test outcomes to the central unit just once and there is no need for regular information updates.

2.2.6 Risk Management Frameworks

Authors in [133] and [66] adopt continuous Hidden Markov Models (HMM) to evaluate system risk based on predefined qualitative categories (e.g. assets, facets) and a weighted

sum approach to compute the overall risk. However, they rely to a large extent on a priori knowledge about the system (e.g., event occurrence probability, impact of the observations, initial state distribution, state transition rate matrix, etc.). Furthermore, a HMM is required for each asset's facet, which makes the system's representation entangled and sensitive to slight departures from the envisioned initial models.

The works in [7] and [127] are concerned with risk modeling in critical infrastructures. The former aims at modeling the security properties of interdependent systems and measures their risk levels and assurances. The latter envisions a service decomposition graph for risk assessment and an online monitoring tool of three risk parameters.

None of the previous frameworks exploit clustering of the risk features as a visual aid for the user, who can then manually alter the modeling of the system's risk factors. None provide easily interpretable granules for shaping the risk features or simple adaptation mechanisms for their associated weights. They are thus not truly evolving, human-centric platforms.

2.2.7 Evolving Fuzzy Clustering

Clustering is one the most active and vibrant research topics within knowledge discovery, pattern recognition and data mining. This rapidly developing framework is often use to communicate underlying structural properties of usually large data repositories to the system analyst in a simple and appealing fashion. In the context of the risk management framework outlined in Chapter 7, the visualization module exploits the benefits of clustering the sensor nodes in the risk feature space to provide the human expert with valuable schematic insights into the prevalent WSN dynamics at any point in time and thus enable him to make risk-aware decisions that will definitely impact the system under consideration in the near future.

Within the broad framework of clustering, *fuzzy clustering algorithms* [139] play a fundamental role given their intrinsic ability to capture the partial belongingness of any object (data pattern) to a knowledge structure (cluster) by means of numerical membership grades. They allow efficient processing and interpretation of borderline patterns and highly overlap-

ping concepts that occur across nearly all domains of everyday life.

The early developments in the fuzzy clustering arena were centered around stationary data repositories, thus leading to the emergence of "offline" unsupervised learning methods that were unable to cope with real-time data streams. To accommodate to these novel, dynamic, data-unbounded and challenging environments, researchers have actively pursued the creation of "online" (fuzzy) clustering methods. This is particularly notorious in the ubiquitous knowledge discovery and data stream mining fields. However, these approaches are not entirely flexible, as their are mainly concerned with a parametric rather than an actual structural adaptation to the continuous data flow.

The term *evolving intelligent systems* [3] was recently coined to designate a sort of intelligent algorithms that not only modify their parameters but also their structural underpinnings over time in order to better reflect the current reality present in the data stream. In the case of an *evolving fuzzy clustering architecture* [118], this implies that the current cluster distribution is continuously modified by means of a dynamic cluster formation process in which the discovered knowledge structures are created, merged, split or deleted accordingly. Many of these volatile clustering architectures come disguised as *evolving fuzzy rule-based systems* [13] if the rule antecedents describe cluster prototypes in the feature space.

Our evolving fuzzy clustering scheme in Chapter 7 is neither rule-based nor incremental (i.e. data-sample-based) but it splits the data stream into multiple snapshots and operates with each piece individually, although providing a sense of continuity across the entire data stream. We are not aware of any other algorithm of its kind that has been applied before to the risk visualization in a WSN. Furthermore, the internal cluster representation follows the shadowed set paradigm (see Sect. 2.1.3), thus becoming the first-ever evolving shadowed-set-based clustering machinery.

2.3 Conclusions

By surveying the bibliography throughout this chapter, we have made clear how the proposed contributions of this thesis (stated in Sect. 1.3) advance the state of the art in sensor placement and relocation by mobile actuators, combinatorial optimization problems, metaheuristic algorithms inspired by communities of living animals, fault identification in diagnosable systems, risk management frameworks and evolving fuzzy clustering techniques, thus ultimately contributing to the feasible, timely development of fault-reactive sensor networks.

More concretely, we have formulated for the first time the novel problems of carrier-based focused coverage formation and coverage repair in WSRNs and put forth efficient algorithms to solve them. Additionally, we have improved the performance of existing metaheuristics in the field of system-level fault diagnosis with two competitive schemes. Finally, an evolving risk management framework for WSNs has been formulated and applied to a simulated Critical Infrastructure Protection scenario.

A thorough description of the thesis contributions is found in Chapters 3 - 7.

CHAPTER 3

CARRIER-BASED FOCUSED COVERAGE FORMATION

In this chapter, we formally introduce the *focused coverage formation* (FCF) problem in the context of a WSRN. Then, a distributed (and strictly localized) algorithm that elegantly solves FCF is described. We shed light on some of its analytical properties before empirically examining its performance through an extensive simulation study.

3.1 Introduction

Coverage [30] is a functional basis of any WSN due to its inherent surveillance goal. It is measured by the area under the supervision of the network. The larger the coverage a WSN exhibits, the better the service it provides. Li et al. [87–90] introduced a new type of problem, coined as *FOCUSED coverage* (F-coverage), characterized by the need of monitoring a given point of interest (POI) and its vicinity. Optimal F-coverage has maximal *coverage radius*, defined as the radius of the largest hole-free (sensing coverage) disc centered at the POI, i.e., the minimum distance from the POI to uncovered areas. Illustrative F-coverage scenarios encompass surveillance and tracking services around a critical region or infrastructure such as battalion headquarters or nuclear power plants, for warfare and homeland security purposes, respectively. Renewed international efforts are being carried out nowadays in disaster man-

agement operations, where an earthquake epicenter or an active volcano crater deserve to be constantly watched in order to minimize the tragic loss of human lives. In any of the previous cases, the deployed sensor network monitors any suspicious event in the proximity of the POI and reports to a certain base station. The distance from a spotted event to the POI reflects the degree of potential damage of the event. From the base station, the command and control unit retrieves real-time sensor readings and makes timely reactive decisions.

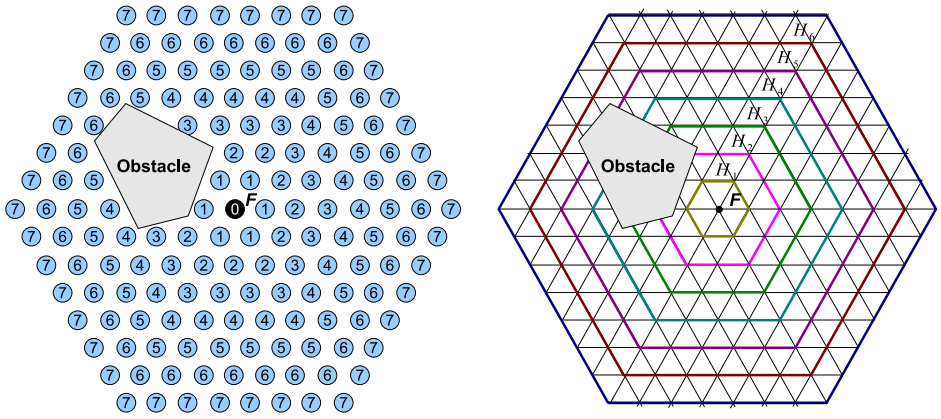
As stated in Chapter 2, no study has been found in literature that is concerned with F-coverage formation via mobile carrier agents. We present the first protocol of its type. It involves robots repeatedly reloading sensors and placing them in the environment. A detailed problem statement follows.

3.1.1 Problem Statement

We consider an unknown two-dimensional environment where the location of a POI, denoted by F, is given. One or a few mobile robots, each with limited carrying capacity, are available for carrying and deploying sensors. Sensors are collected at one or more fixed locations, called *base points*. Robots have to be loaded with sensors and enter the environment from there. After finishing with currently loaded sensors, they return to the base points for reloading and continue sensor placement.

Robots are aware of their own locations and able to detect nearby obstacles. For the sake of simplicity, we will assume that robots and sensors have the same communication radius R_c, although in practice the former are able to communicate way further than the latter. Another simplifying assumption for modeling purposes is that all sensors share the same sensing radius R_s. Once deployed, they periodically exchange '*hello*' messages that carry their location information, thus becoming localized to the robots that place them. This message is a basic networking technique for e.g. routing [19] and activity scheduling [59]. We assume $R_c \geq \sqrt{3}R_s$ as in the literature [12, 25, 26, 88]. Sensors may fail at any time.

The goal is to develop a carrier-based sensor placement algorithm that yields a sensor



(a) An optimal F-coverage (b) Layout of a TT graph

Figure 3.1: F-coverage problem envisioned as a vertex coverage problem over a TT graph.

network surrounding F in hexagonal layers and with an equilateral triangle tessellation (TT) layout of nodal separation $\sqrt{3}R_s$, as shown in Fig. 3.1(a). This particular type of sensor arrangement is desirable because it guarantees connectivity and optimal hexagonal F-coverage [87–89]. The sensor placement problem is therefore turned into a vertex coverage problem on a TT graph containing F as center, as shown in Fig. 3.1(b), where hexagonal layers are indexed by their graph distance to F and denoted by $\mathcal{H}_{index}$. In Fig. 3.1(a), the number inside each node indicates the index of the hexagon where it resides. We elaborate on the topological features of the TT graph in Sec. 3.2.

3.2 Equilateral Triangle Tessellation

An equilateral triangle tessellation (TT) is a planar graph composed of congruent equilateral triangles, as shown in Fig. 3.1(b). With a common orientation, say north, each sensor is able to locally compute a unique TT graph of edge length l_e with F (the POI) as center. Denote the TT graph by G_{TT}. In our work, l_e is set to $\sqrt{3}R_s$. It is because we want to finally locate sensors on vertices of G_{TT}, and this particular edge length ensures connectivity and minimizes sensing range overlapping [10, 98, 153].

Let SP(u, v) represent the shortest path connecting two vertices u and v in G_{TT}. Then the

TT distance between u and v is defined as the number of edges in $\mathrm{SP}(u, v)$ and referred to as $|\mathrm{SP}(u, v)|$. There are $6i$ vertices with equal TT distance i to F in G_{TT}. They constitute a distance-i hexagon, denoted by $\mathcal{H}_i$. These $\mathcal{H}$ hexagons are concentric to F. As proved in [88, 89], deploying sensors along $\mathcal{H}$ hexagons will produce optimal or near optimal F-coverage. The total number $v(i)$ of vertices enclosed by $\mathcal{H}_i$ (inclusive) is

$$v(i) = 1 + \sum_{q=1}^{i} 6q = 3i(i+1) + 1 \quad . \tag{3.1}$$

3.3 Carrier-Based Focused Coverage Formation

In this section we present *Carrier-Based Focused Coverage Formation* (CBFCF), a new protocol that uses mobile robots to lay static sensors and build optimal F-coverage around the POI in a progressive fashion. This algorithm is composed of two major components which are run in parallel: *reactive advertising routine* (RAR) and *iterative sensor placement* (ISP).

Recall that sensors are to be placed on the vertices of TT graph (see Sec. 3.1.1). RAR runs locally on each deployed sensor on demand. Its objective is to publish along the network border any empty vertex induced by node failure as well as any available vertex beyond the current boundary of the network. ISP runs in a distributed manner on sensors and robots spanning multiple iterations. At each iteration, robots enter the environment from base points, place sensors at empty vertices and then return to fetch sensors for a new iteration. Due to asynchrony, different robots may be in different iterations at any moment in time.

CBFCF terminates when ISP stops iterating on all robots, namely, when either all available sensors have been deployed or the environment is fully covered. At this point, all robots have returned to their base points. We elaborate on the RAR and ISP routines below.

3.3.1 Reactive Advertising Routine (RAR)

Each sensor locally maintains an *empty vertex list* (EVL) once being deployed. The list initially contains the six adjacent vertices in the TT graph and gets dynamically updated as neighboring sensors are discovered (by hearing a '*hello*' message from them) or identified failed (by missing a pre-defined number of successive '*hello*' messages from them). For a detected failed neighbor at vertex v, a sensor transmits a *failure notification* message containing the location of v.

The failure notification is forwarded outward, away from F, along a single path by sensors following the Greedy-FACE-Greedy (GFG) routing principle [19]. In this process, greedy forwarding is implemented by forwarding the message from hexagon to hexagon in the ascending order of their indices. The message stops, after traversing the entire border of the network (i.e., the perimeter of the outer face), at a border node having previously received the packet from the same direction. This node then forwards the message along the network border by one more round of face traversal (needed to locally identify whether a node lies at the network border or not). Then, all border nodes get v from the message and insert it into their local EVL.

At most six copies (originated from the different neighbors of the failed sensor) of the same failure notification message may circulate simultaneously. To save message retransmissions, each node forwards the same failure notification only once. By RAR, the local EVL of border nodes contains all the empty vertices inside the network in addition to the locally identified adjacent ones outside the network.

Figure 3.2 illustrates the RAR routine. The failed node is marked by a cross and dashed arrowed lines imply the transmission paths of the failure notification messages. For example, node M detects the failure and sends a notification message greedily outward, from low-index hexagon to high-index hexagon. This message reaches border node S_1 and then propagates along the network border in FACE routing mode. It stops at border node S_2 because it had already received and forwarded the same failure notification originated at F. Upon receiving

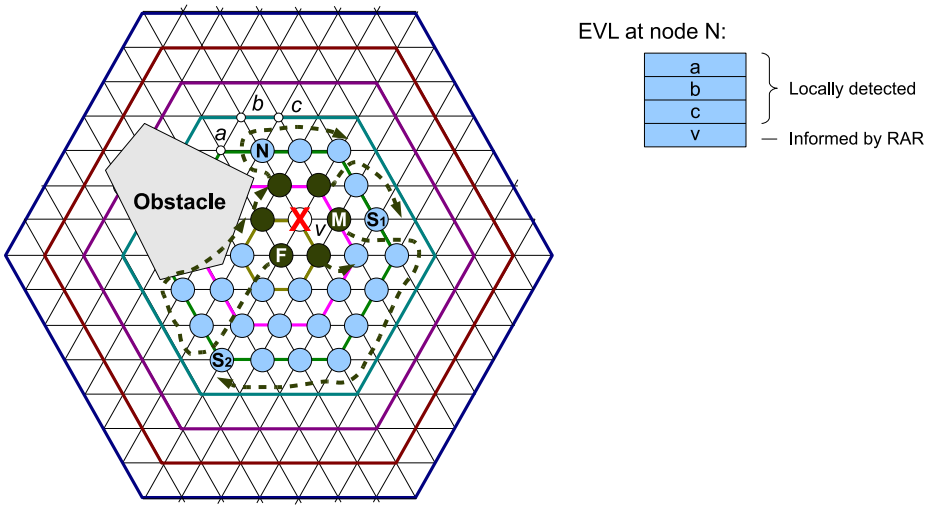


Figure 3.2: Reactive advertising routine (RAR)

this message, S_2 starts the second round of border traversal by another failure notification.

3.3.2 Iterative Sensor Placement (ISP)

We will now unveil ISP by focusing on a single iteration with respect to an arbitrary robot B. From the local perspective, the ISP iteration starts with B (loaded with sensors) entering the environment and ends with B returning to its base point.

Once entered the environment, the robot *marches* toward F as follows: it moves greedily along the straight line and, when obstructed by an obstacle, it rotates around it following a predetermined direction, e.g., clockwise, until greedy movement can be resumed. While marching, it periodically transmits a beacon message containing the target location. Upon receiving the beacon message, an early deployed (possibly by a different robot) sensor replies with a '*hello*' message. B stops marching and beaconing as soon as it receives a '*hello*' message or it reaches F. In the former case, it enters a *search phase* where it looks for an empty vertex to drop a sensor and subsequently engages a *migration phase*, in which it silently marches (no beaconing) to the discovered vertex to lay a sensor.

By B arriving at a location v, either F or a discovered vertex, another robot may possibly be also approaching v for sensor placement or a sensor may have already been deployed by

another robot. To avoid multiple placement, B waits at v and beacons on a periodical basis, up to a maximum number of times, before dropping a sensor there. During this period, it identifies one of the following cases and acts accordingly.

(1) **Void case**: there are no neighboring sensors. This can be determined by hearing no sensor '*hello*' message.

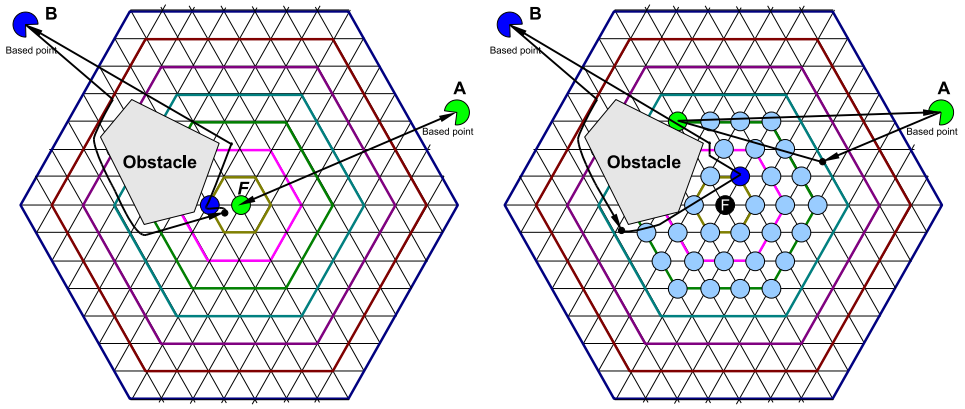
(2) **Occupancy case**: a sensor has already been placed at v. It takes place when hearing a '*hello*' message originated at position v.

(3) **Competition case**: another robot is attempting to place a sensor at v. We realize about that after hearing a robot beacon message containing v as its target location.

(4) **Ordinary case**: any case different from the ones above.

In the void case, B resumes marching toward F with periodical beaconing, as v is isolated from the network. In the occupancy case, B enters a *search phase* for finding a different vertex. In the competition case, it competes with that robot. The one with smaller ID wins and drops a sensor. The loser will soon receive a '*hello*' message from v and run into the occupancy case. This competition may happen several times, involving different robots, before any sensor could be actually dropped. In the ordinary case, B drops a sensor at v and starts beaconing so as to find another empty vertex.

Figure 3.3 portrays the last two cases, where robots A and B both have cargo capacity of one (i.e. the ability to carry at most one sensor) and travel along the thick arrowed lines. In Fig. 3.3(a), they meet at F and compete for sensor placement. A wins the competition –because $\mathrm{ID}(A) < \mathrm{ID}(B)$– and drops a sensor. Loser robot B later on receives the '*hello*' message from the sensor laid by A. It finds out, through this sensor, another deployment spot adjacent to F and moves to drop a sensor there. In Fig. 3.3(b), B hits the network border and discovers (and reserves) a vertex inside the network. This vertex is available because the sensor previously occupying that position failed and the neighboring sensors advertised it along the network



(a) Competition at F (b) Ordinary case

Figure 3.3: Iterative sensor placement (ISP)

border (see Fig. 3.2). At the same time, A hits the network border and learns about an empty vertex on the outermost hexagon layer. It does not discover the one already reserved by B.

The robot returns to its base point (thus finishing the current ISP iteration) if it runs out of cargo (i.e. all loaded sensors have been deployed) or no empty vertex can be found. Otherwise, it will continue to place sensors according to the joint search-migration strategy. After finishing an ISP iteration, a robot may or may not start a new iteration depending on whether or not it depleted its sensor load in the previous one. Figure 3.4 gives the sequence diagram of a complete ISP iteration. Central to ISP is apparently the search phase, which we will address separately in the next section.

3.3.3 The Search Phase – an ISP Sub-procedure

In the search phase, the robot B discovers an empty vertex that is adjacent to the already established network (for connectivity purpose) and located on a hexagon of smallest index (for coverage radius maximization). Figure 3.5 shows the sequence diagram of this phase.

To start a search phase, B transmits a *search* message carrying its current location outwards, away from F, and expects a *reply* message containing search result. If no reply arrives within a certain period of time, B will restart the search (doubling the response timeout) and

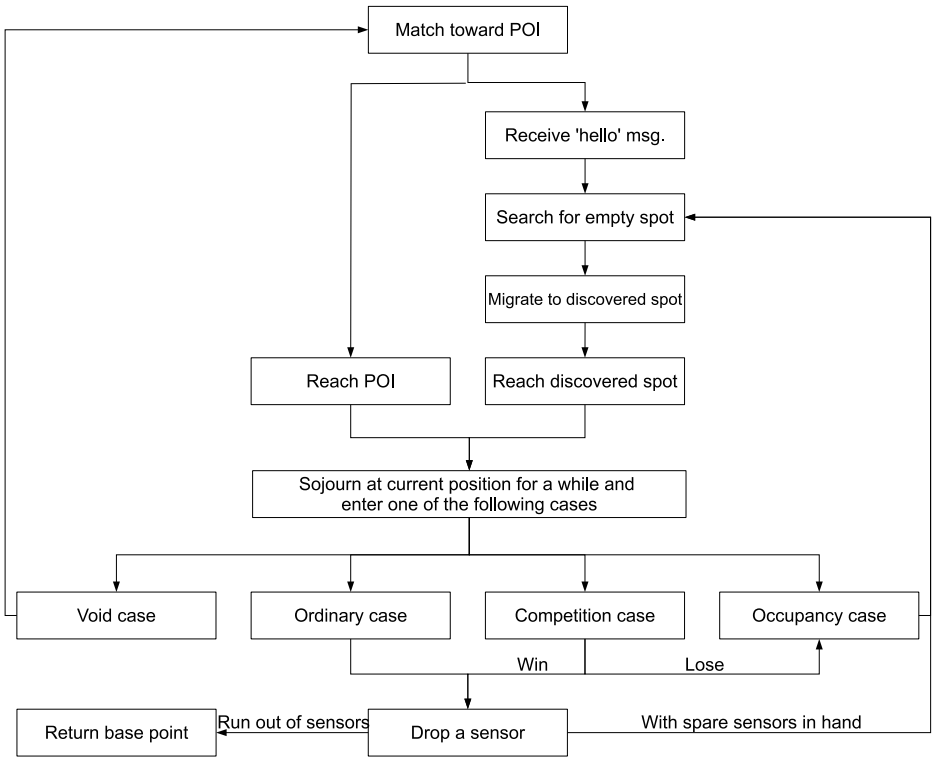


Figure 3.4: Flow chart of an ISP iteration

ignore any late reply; otherwise, it routes an ACK message to the sender (whose location is embedded in the reply message) by GFG [19].

The search message is routed away from F by sensors following the GFG principle as the failure notification message did in RAR. It stops at a border node S that received the packet for the second time and from the same incoming direction. This node identifies itself as the *search agent* of B and continues the search on its behalf.

The search agent S forwards the search message of B along the network border by face routing. This can be definitely done because S has knowledge of the outer face. The message picks up the location of the empty TT vertex v nearest to B among those with lowest hexagon index and stored on each forwarding node. Once the message gets back to search agent S, it learns about the search outcome v and circulates a *reservation* message along the network border so that v is no longer recommended to any other robot. By doing this, vertex contention is properly taken care of.

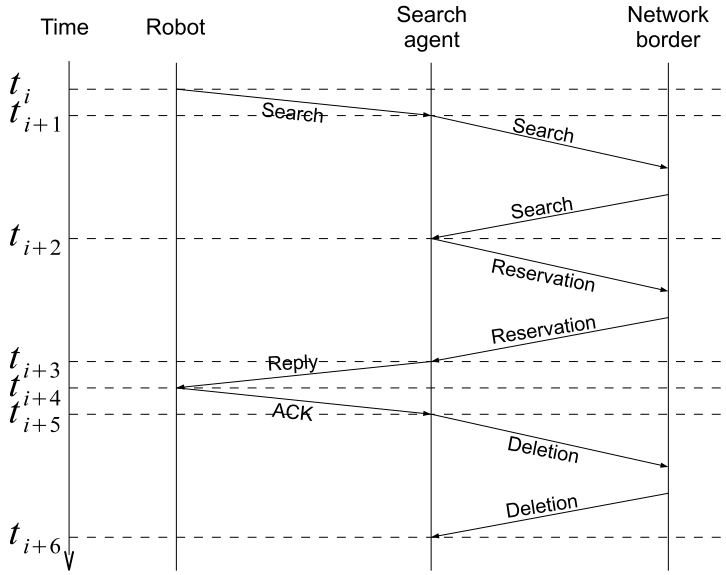


Figure 3.5: Sequence diagram of the search phase

If a border node has confirmed a reservation for v to another robot by the time of receiving the reservation message from S, it will set a flag in the message. By checking the flag in the returned reservation message, S knows whether the reservation succeeded. In case of failure, it restarts the search immediately. Should the reservation be confirmed, it notifies B about the outcome (via a *reply* message) which is routed according to GFG and waits for an ACK. On the arrival of the ACK from B, node S erases the information of v from the network border (i.e. the EVL of all border nodes) by another round of border traversal through a *deletion* message. If no ACK arrives within a certain period of time (e.g., due to node failure), it releases v from reservation by sending a *cancellation* message along the network border.

Although the ISP subprocedure was devised in a systematic fashion so as to minimize the likelihood of inappropriate vertex contention and inconsistent information in the local memory of the sensors, its reservation scheme can still give rise, under particular conditions, to a problematic circumstance which we have baptized as *reservation deadlock*. In Fig. 3.6, two robots are located at nodes labeled a and b and both issue (asynchronously) their own *search* message in an attempt to find an empty spot for sensor placement. Each message finds its way towards the search agent node that will take care of it. The best location currently

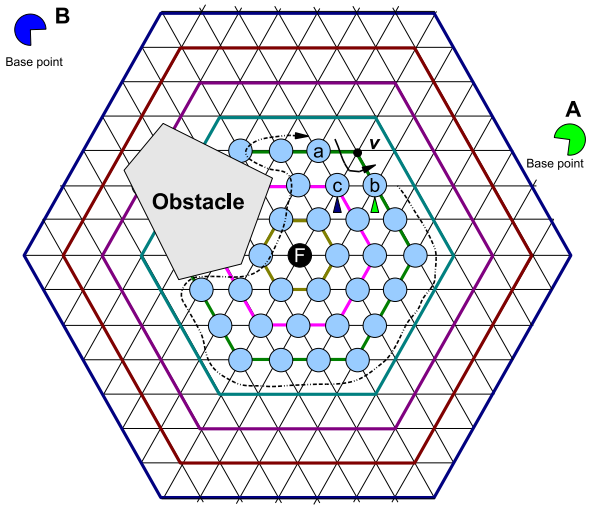


Figure 3.6: Illustration of a reservation deadlock occurring at vertex v.

available is vertex v and both search agents properly learn about it and get ready for reserving that spot for the robot whose *search* message they are responsible for.

One will realize from the figure that none of the reservation messages will come back with a favorable outcome (reservation success), as their face traversals mingle at points a and b respectively and each message flag will be set to true as an indication that the contended spot has been reserved for somebody else. In that way, both robots will actually get stuck in a "deadlock" status where none of them will be able to get out. Subsequent *search* messages emitted by the robots will endure the same fate as the previous ones.

Whenever a reservation deadlock surfaces (Fig. 3.6), it will be broken by the closer actuator to the spot (among all contestant robots). Ties are won by the robot with lowest ID.

3.4 Optimization

Suppose that two robots A and B are augmenting focused coverage around F. They get into the monitoring region from dissimilar base points and march toward F asynchronously. They discover two empty vertices u and v (both caused by node failure) inside the already-established network after arriving at positions a and b, respectively. Then they will choose

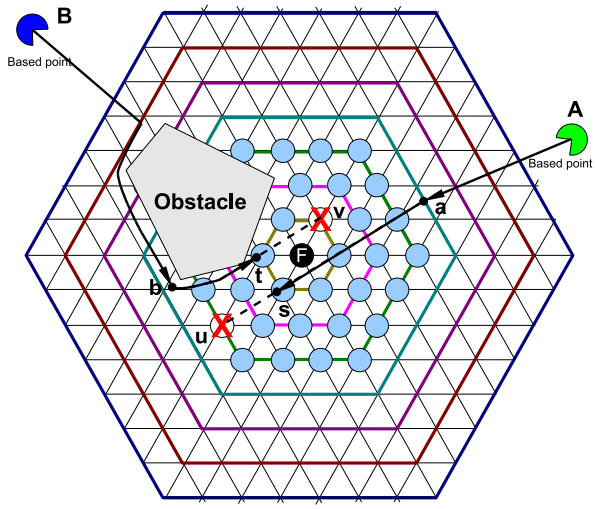


Figure 3.7: Robot task exchange (RTE)

these two vertices as their target locations for sensor placement, i.e., A heads to u and B heads to v. Both robots meet each other on their way. The meeting gives them a chance to exchange sensor placement tasks, which may be beneficial in terms of distance traveled or coverage repair latency.

Assume that, at the meeting time, A is located at vertex s and B finds itself at t, as shown in Fig. 3.7. Notice that v is closer to F than u. Suppose B is a slower robot that A. If A and B stay with their original sensor placement tasks, coverage augmentation will not be achieved soon (in other words, coverage radius will not be immediately increased) even if A quickly reaches u and drops a sensor there, since v will still remain empty for a while due to the slow motion of B. Therefore, the task exchange between A and B will reduce coverage augmentation delay. Even if the two robots are moving at the same speed, in some cases the task exchange may still be desirable because it will lead to reduced moving distance for both of them.

Motivated by the above analysis, we introduce *robot task exchange* (RTE) as an optimization technique for CBFCF. The idea is summarized as follows: whenever two robots A and B encounter each other, by local communication they will exchange their sensor placement tasks should the exchange prove beneficial to either deployment latency or distance traveled. If more than two robots meet together, RTE is performed on a pairwise manner until the task

assignment becomes stable among such robots.

3.5 The Algorithm

CBFCF's workflow during its distributed and concurrent execution on sensors and robots is displayed in Algorithms 1, 2 and 3.

Algorithm 1 Reactive Advertising Routine (RAR): runs on sensors

Input: incoming HELLO messages from 1-hop neighbors
Output: FAILURE_NOTIFICATION message sent to network border

```
1:  EVL ← {coordinates of all 1-hop neighbors};
2:  while true do
3:      if incoming HELLO message from neighbor ⟨i, j, k⟩ arrives then
4:          EVL ← EVL \ ⟨i, j, k⟩;
5:      end if
6:
7:      if missed HELLO message from neighbor ⟨i, j, k⟩ T times then
8:          EVL ← EVL ∪ ⟨i, j, k⟩;
9:          send( FAILURE_NOTIFICATION, ⟨i, j, k⟩ );
10:     end if
11:
12:     if incoming FAILURE_NOTIFICATION message from neighbor ⟨i, j, k⟩ arrives then
13:         if I am a border node then
14:             EVL ← EVL ∪ ⟨i, j, k⟩;
15:         end if
16:         if this message not received before from the same direction then
17:             send( FAILURE_NOTIFICATION, ⟨i, j, k⟩ );
18:         end if
19:     end if
20: end while
```

3.6 Analysis

Some analytical insights on the correctness and reliability of the developed protocol are ventilated in this section. In particular, we elaborate on its global termination, pinpoint technical conditions leading to abnormal behavior and put forward the required algorithmic changes to counter them.

Algorithm 2 Iterative Sensor Placement (ISP): runs on robots

Input: location of *POI*, location of base point BP, nrSensorsAtBP, nrSensorsLoaded
Output: bi-connected layout around the *POI*

```
1:  target ← POI; marching ← true; beaconing ← true;
2:  while target ≠ ∅ AND nrSensorsAtBP > 0 do
3:      if nrSensorsLoaded == 0 then
4:          [nrSensorsLoaded, nrSensorsAtBP] ← replenishSensorsAtBP();
5:          target ← POI;
6:      end if
7:      if marching then
8:          march(target);
9:      end if
10:     if beaconing then
11:         beacon(current location);
12:     end if
13:     if incoming HELLO message from sensor ⟨i, j, k⟩ arrives then
14:         marching ← false; beaconing ← false;
15:         target ← send( SEARCH, current location );
16:         marching ← true;
17:     end if
18:     if target reached then
19:         case ← identifyCase();
20:         if case == VOID then
21:             marching ← true;
22:         end if
23:     else if case == OCCUPANCY then
24:         target ← send( SEARCH, current location );
25:         marching ← true; beaconing ← true;
26:     else if case == COMPETITION then
27:         winner ← competeWithRobots();
28:         if winner then
29:             nrSensorsLoaded ← dropSensor();
30:         end if
31:         marching ← true; beaconing ← true;
32:     else                                          ▷ case == ORDINARY
33:         nrSensorsLoaded ← dropSensor();
34:         marching ← true; beaconing ← true;
35:     end if
36: end while
```

Algorithm 3 Iterative Sensor Placement (ISP): runs on sensors

Input: incoming SEARCH message from a nearby robot
Output: the target vertex for the robot

```
1:  while true do
2:      if incoming SEARCH message from robot at ⟨x_R, y_R⟩ arrives then
3:          send( SEARCH, ⟨x_R, y_R⟩ );
4:      end if
5:
6:      if incoming SEARCH message from neighbor ⟨i, j, k⟩ arrives then
7:          if I am a border node then
8:              if this message not received before from the same direction then
9:                  send( SEARCH, ⟨x_R, y_R⟩ );
10:             else                    ▷ node S is the search agent for this SEARCH message
11:                 target ← reserveSpot( ⟨x_R, y_R⟩ );
12:                 send( REPLY, ⟨x_R, y_R⟩ );
13:             end if
14:         end if
15:     end if
16:
17:     if (incoming ACK message from robot at ⟨x_R, y_R⟩ arrives) OR
18:       (no ACK from robot at ⟨x_R, y_R⟩ arrives after timeout T) then
19:         cancelSpot( target, ⟨x_R, y_R⟩ );
20:     end if
21: end while
```

Definition 3.1. *Persistent communication failure (PCF) is the inability of a sensor to transmit a particular type of message for an indefinite period of time.*

The following results are derived under the assumption that no node suffers from PCF. Then we unveil how PCF jeopardizes CBFCF's normal operation and describe the algorithmic modifications needed to overcome this setback.

Lemma 3.1. *ISP at the robot side terminates in finite time.*

Proof. From Sec. 3.3.2, one realizes that doubling the response timeout ensures that a *reply* message will be eventually received, thus finishing the search phase in finite time and enabling sensor dropping to occur. In the end, after multiple ISP iterations, either the robot's base point runs out of sensors to deploy or a *reply* message indicates that no TT empty vertex could be found. This marks the ISP local termination for a robot, after which it goes back to its base point (if not already there) and shuts down CBFCF's execution. □

Lemma 3.2. *CBFCF globally terminates in finite time.*

Proof. The RAR module continuously runs on sensors. Its deactivation is not desirable so as to report prospective sensor failures to the network border. ISP runs on robots and sensors. Local ISP termination in finite time for a robot is guaranteed by Lemma 1. Global termination follows from local termination of all robots. □

The ISP search phase might never terminate if the sensor feeding the robot with the search outcome has PCF. A simple way of avoiding this pitfall is by having the robot march to POI (with beaconing) after missing a number of replies.

PCF also strikes RAR's ability to publish damaged spots to the network border. Should all neighbors of v in Fig. 3.2 undergo PCF, then its coordinates remain unknown to the boundary nodes and the coverage radius is reduced. Fortunately, this scenario is not likely unless an event affects a certain geographical area of the monitoring field.

Moreover, if at least one unit in any of the reporting paths endures PCF, the network is cast into an *inconsistent state*, for some nodes will learn about v's location and others will

not. One solution is the retransmission of the *failure notification* packet with frequency λ, hoping that new hops could be contacted as the network topology evolves in order to bypass the PCF-tainted node.

Let L be the length (number of hops) of a reporting path and ϕ the probability of sensor failure in the field. A damaged unit was detected at time t_0 and CBFCF finishes at time t_f. Expression (3.2) models the probability that an inconsistent state arises because of that PCF-permeated path, where $\lfloor \cdot \rfloor$ is the floor function.

$$1 - \left[(1-\phi)^L\right]^{\lfloor \frac{t_f - t_0}{\lambda} \rfloor} \tag{3.2}$$

Two avenues are envisioned to minimize the lack of knowledge integrity in the network: either we increase the periodicity of the failure notifications λ (and pay a steep price in terms of communication overhead) or prolong the overall execution time t_f. The latter is preferable and could be attained by simply dropping the global termination requirement. This means that robots still having spare sensors, after completing their local deployment task and settling in their base points, are allowed to occasionally march to the POI until they touch the network border, at which point they become aware of any post-deployment failure and tend to it.

3.7 Experimental Results

This section assesses CBFCF's performance from diverse standpoints. Since it is the first known protocol in literature that addresses the problem of *carrier-based sensor placement by mobile robots in F-coverage scenarios*, we cannot contrast CBFCF to any other approach. Therefore, the empirical analysis aims at providing a comprehensive picture of the advantages and limitations of our method.

3.7.1 Simulation Metrics

To estimate CBFCF's behavior in terms of deployment latency and resource expenditures, the following performance indicators will be employed:

Convergence Time (T, I) T is the number of time units required for the algorithm to finish. Upon termination, CBFCF yields a uniform TT layout around the POI with maximal coverage radius using the number of sensors available. I is the average number of iterations executed by a robot before it either ran out of sensors in its base point or the entire TT was built.

Mobility costs (M, V, MPI) Total energy spent through the algorithm's execution can be quantified from two perspectives: mobility costs at the robot side and transmission costs incurred by both sensors and robots. For the former, we measure the average distance traveled per robot (mileage M) and the average number of times a robot restarts its motor (number of moves V). This is because starting a still motor consumes quite a large amount of energy. In our implementation, we will assume that a robot keeps its engine on while waiting for an ensuing *reply* message to its previously emitted *search* request. Should it not arrive on time, the robot turns its mobile platform engine off to save energy until a newly discovered spot is reported afterwards. The average distance walked by a robot per iteration (mileage per iteration, MPI) is also recorded. Due to the asynchronous nature of the sensor placement task, each agent may be in a distinct iteration at a given time.

Transmission costs (RM, SM) The average number of messages generated by a robot (sensor) is denoted by RM (SM). They quantify the energy usage for communication purposes per robot (sensor).

Robot collision (C) Collision does not necessarily mean that two robots physically crash at a geographic location, but that they are sufficiently close to one another (our proximity threshold has been set to $l_e/10$, i.e. a fraction of the internodal distance in the TT graph).

Collision stems from asynchronous robot movement during sensor deployment and must be duly reported given the potential communication failures brought about by the radio signal interference in the physical layer of the colliding devices. For the same collision not to be reported multiple times, we only allow the robot with the lowest ID in the pair to keep track of it. The robot only counts the collision if it either takes place at a new spot in its local collision list or occurred at the same location "a while" ago. In our implementation, the time span between two collisions detected at the same coordinates is equal to three time units. The C indicator stands for the average number of collisions per robot.

We chose to quantify CBFCF's power consumption features in terms of distance traveled and messages transmitted rather than turning to a particular energy model (e.g. path loss [125]). By doing so an unbiased, model-independent picture of the energy expenses of the proposed algorithm is presented.

3.7.2 Simulation Setup

CBFCF has been implemented with JBotSim simulator[1]. Experiments were conducted on a Compaq Presario CQ 50 Notebook PC with an AMD Athlon Dual-Core QL-60 processor at 1.90 GHz and equipped with 3.0 GB of RAM running Windows Vista Home Premium under a 32-bit architecture. JBotSim relied upon JDK 1.6.0_18 framework.

Deployment of static sensors was simulated over a square region of variable size whose center is regarded as the POI. Initially, the area is empty and robots lie at predefined base points. Stationary units have sensing radius $R_s = 20$. Both types of nodes share communication radius $R_c = 2 \cdot R_s$. Robot speed varies from 0.05 to 0.2 per time unit and its cargo capacity equals five sensors, which is reasonable and realistic for a wide array of scenarios. The internodal distance l_e in the TT graph was set to $= \sqrt{3} \cdot R_s$.

In the first experimental setting, we study CBFCF's performance with a single mobile robot and varying network sizes (from $\nu(1) = 7$ to $\nu(10) = 331$, check Eqn. (3.1)) while

[1] Available at http://jbotsim.sourceforge.net/

keeping the field size unchanged (700×700). No sensor failures occur.

A second group of trials sheds light on the impact of the robotic fleet size over the algorithm's demeanor. CBFCF aims at laying $\nu(7) = 169$ sensors across a 500×500 field while gradually raising the number of actuators from 2 to 5. The efficiency of the RTE optimization module in Sec. 3.4 will be gauged. A failure-free scenario is envisioned again.

Our last ensemble of simulations embraces the likelihood of sensors failing at different rates due to manifold reasons. The focus is now to measure CBFCF's adaptability and robustness when handling these unpredictable yet pragmatic situations.

Values reported in the upcoming plots are the means of the performance metrics in Sec. 3.7.1 and their 95% confidence intervals. Means are computed over 50 independent runs so as to minimize the bearing of CBFCF's asynchrony.

3.7.3 The Experiments

In the following, we elaborate on the set of comprehensive simulations carried out to validate the feasibility of our carrier-based sensor placement algorithm.

Fixed Area Size; Variable Network Sizes

A single mobile robot builds a F-coverage formation around the center of a 700×700 field. It departs from base point located at (700; 350) and uniformly advances one distance unit per time unit. No sensor gets out of order during CBFCF's execution.

Figures 3.8(a) – 3.8(g) display CBFCF's demeanor as the number of sensors n to be deployed in the surveillance area grows. Most of the charts reflect the anticipated trends of monotonically increasing relationships of the performance measures in Sec. 3.7.1 to n. Confidence intervals are moderate (even negligible in many cases), not exceeding 25% of the mean of the indicator under discussion. This speaks highly about the preciseness and reliability of the algorithm despite the stark non-deterministic behavior caused by its distributed nature.

Figure 3.8(a) reveals a quadratic dependence of the convergence time T on n, for it takes

longer to deploy a greater number of sensors. Interestingly, a significant share of the time expenses (35% – 41%) is due to sensor reloading, i.e. robots ran out of sensors and return to their base points to replenish them.[2] Additionally, robots spend roughly 12% of the time standing by until receiving feedback from the sensors in the form of *reply* messages. The rest of the time is devoted entirely to laying sensors at the suggested coordinates.

The sublinearity between the average number of iterations per robot I and n is exposed in Fig. 3.8(b). It is easy to see that the number of roundtrips a robot performs depends on its capacity and the number of sensors available at its base point. Since this configuration is identical for each run of the algorithm, the standard deviations are zero and thus omitted.

The average robot mileage M also rises quadratically with n, as shown in Fig. 3.8(c). The greater the network size, the higher the mobility costs per robot, as more TT vertices are to be visited. From $n = \nu(7) = 169$ on, the confidence intervals get wider compared to those of small networks. This is direct consequence of the bigger role asynchronous communication plays as the network becomes larger, for there is a broader spectrum of multi-hop paths through which info on the next recommended spot is conveyed. Depending on asynchronous message arrival, the actuator may choose sometimes shorter, sometimes longer routes (straight to its target) and this causes M to fluctuate more noticeably. Likewise T, 28% – 46% of the average mileage goes to sensor fetching from base points.

Because M and I climb up with n, the mileage per iteration ratio MPI remains bounded between 600 and $1,000$ distance units, as portrayed in Fig. 3.8(d). The nearly asymptotic MPI behavior along an extensive range of network dimensions leads to the following deployment principle: instead of purchasing a few well outfitted robots to take care of the entire deployment process individually, let us acquire a bigger fleet of less powerful agents and have them take turns, each running for a whole iteration. This proves beneficial since the corporate workload is fairly balanced for a broad range of network sizes.

As to the transmission costs, Figs. 3.8(e) and 3.8(f) make clear that a linear increase must

[2]Deployment times reported by well-known carrier-based sensor placement protocols in literature are often shorter because they do not acknowledge the limited cargo capacity of the mobile actuators.

be expected in the average number of messages for both robots and sensors. The latter undergo a steeper incline, for CBFCF actively exploits sensor-sensor and sensor-robot communication to make deployment decisions. This scheme is several orders of magnitude cheaper than locomotion, upon which approaches like GRG [87–89] heavily rely. Beacon messages for robots and *hello* messages for sensors arise as the most usual types of emitted messages (32% – 51% for the former and 28% – 57% for the latter). This overhead can be alleviated by reducing the beacon frequency but at the expense of a prolonged execution time.

Regarding the packet distribution per sensor, those in the network border have to deal with 60% – 75% more packets than the "inner" nodes. This is because boundary nodes must know the location of all available TT spots and are thus liable for providing accurate information to the robots in their quest for empty vertices. Half of the 10 message types in CBFCF are initiated/routed solely by border nodes. However, this uneven messaging load for border nodes vanishes as CBFCF continues to fill more outer hexagons. In other words, border nodes do not get "worn-out" because newly deployed sensors will take over their role as the coverage radius is expanded.

Bigger networks imply further energy needed to ignite a robot's still motor, as evidenced by the sublinear curve in Fig. 3.8(g). *Reply* messages are missed more often as more sensors are to be deployed, for vertex-contention-related packets circulate along a larger network border, thus taking longer for the robot to learn about the search outcome. This situation can be circumvented by stretching the timeout for *reply* packets at the robot side, yet again delaying the convergence time for CBFCF. Such decision is largely problem-specific.

An interesting phenomenon observed during CBFCF's execution is what we call the *newbie effect*. Recently deployed nodes (*newbies*) can provide imprecise information to the *search* messages triggered by the actuators, for it takes a while before such nodes update their local EVL with *hello* packets coming from neighboring sensors. As a result, they can recommend vertices in their inner hexagon which are currently occupied, thus causing wasteful robot moves and misleading *reservation* (and subsequent *deletion*) messages.

In an attempt to overcome this detrimental event, a node regards itself as newbie until either it receives its first *hello* packet or a prudential time elapses. A newbie suggests no spot to a *search* message conveyed along the network border, just relays it to the next hop. This workaround however does not eliminate the newbie effect, as a node might receive its first *hello* message from a neighbor not in its inner hexagon, thus causing a change in its 'newbie' status yet still maintaining its partially outdated EVL. A more aggressive strategy is required, this time at the robot level.

Whenever a robot drops a sensor, it records its location as the last deployed unit and refuses to accept any *reply* message from that node until a prudential time elapses, needed for the node to catch up with neighbors, if any. This rules out newbies acting as search agents yet still is not a true solution for the problem, for the newbie (generally laid at the network border) could still be queried (after incorrectly changing its status to non-newbie) during a *search* message traversal launched by another search agent unit.

Fig. 3.9(a) reports the *newbie distance ND*, i.e. the distance walked by the carrier robot in response to misleading "best spot" updates from newbie sensors. It grows with the network size as expected, but not monotonically because it is partially attenuated by CBFCF's asynchronous character. Fortunately, in Fig. 3.9(b) we realize that ND is almost negligible compared to M and irrespective of the network dimension. The joint robot-sensor strategy works better in small networks. Overall, $ND \leq 3\%M$.

Multi-Actuator Coordination

Our second ensemble of experiments aims at testing the inter-robot coordination mechanisms that are one of CBFCF's building blocks. Now we confine ourselves to a medium-size field (500×500) in which $n = \nu(7) = 169$ sensors are to be deployed around its POI. The number of actuators will be gradually augmented, starting with two and reaching a maximum of five robots. Selected base points are located at $(500; 250)$, $(500; 500)$, $(0; 0)$, $(50; 480)$ and $(480; 50)$, respectively.

Table 3.1: CBFCF running WITHOUT the RTE optimization technique

Robots	T (10^4)	M (10^3)	V	C
2	2.888 ± 0.30	12.202 ± 1.35	57 ± 12.8	6 ± 0.5
3	1.945 ± 0.45	7.536 ± 0.89	68 ± 15.1	15 ± 1.6
4	1.476 ± 0.35	5.791 ± 1.09	72 ± 11.8	32 ± 8.7
5	1.173 ± 0.22	4.801 ± 0.10	75 ± 2.3	57 ± 12.8

Table 3.2: CBFCF running WITH the RTE optimization technique

Robots	T (10^4)	M (10^3)	V	C
2	**2.106 ± 0.14**	**10.329 ± 2.36**	**47 ± 4.0**	**5 ± 1.5**
3	**1.577 ± 0.39**	**6.440 ± 0.92**	**53 ± 0.1**	**12 ± 0.6**
4	**1.043 ± 0.23**	**3.254 ± 0.58**	**59 ± 0.6**	36 ± 3.1
5	**0.846 ± 0.04**	**2.950 ± 0.14**	**67 ± 10.1**	**54 ± 8.2**

Tables 3.1 and 3.2 lay out the results obtained after running CBFCF without and with the RTE module in Sec. 3.4, respectively. Means and their 95% confidence intervals are displayed. Improvements caused by RTE are highlighted in bold.

Confirming the outcomes of the first experiment, the 95% confidence interval of the mean of each performance metric is enclosed within 30% of its standard deviation, which speaks about a steadfast, robust behavior of the algorithm. Notice the immediate bearing brought forth by the addition of extra robots over the deployment time in both tables. Parallel sensor placement by multiple mobile agents is indeed profitable. Such decline is more remarkable in presence of the RTE optimization strategy (reductions of even 41% of the convergence time were achieved when introducing four robots), thus validating the importance of the dynamic exchange of target locations between neighboring agents when needed.

Mileage M also gets significantly lowered with RTE, for individual robots walk less on average as their population grows. A peak of 38% in distance reduction is observed when five robots cooperate via a set of pairwise location exchanges. The downside is that collisions take place more often as a larger number of robots are available and RTE doesn't seem to have much influence over it, as evidenced by the fourth row of Table 3.2, where 36 average collisions per robot were recorded with RTE versus 32 without it.

The above tables further reveal that when multiple actuators work together, the mean number of moves per robot V ramps up[3], regardless of whether the RTE scheme was enforced or not. The explanation for such fact lies in the reservation deadlock phenomenon previously described. The more robots are in the field, the more *search* messages there will be in the network. Hence, the more likely it is that the different traversal paths along the network border will intertwine with one another, thus causing *reservation* messages to fail and triggering another round of face traversal per fruitless reservation. As a result, the timeouts associated with the *reply* packets at the robot side will go off and thus robots will shut down their mobile engines to save power. From the reported values, one realizes that the efficiency of the contention mechanism for reservation deadlocks devised in Sec. 3.3.2 degrades as the robot population is augmented.

Dealing with Sensor Failures

CBFCF is finally tested in presence of sensor failures. For each experiment, the failure rate ρ is chosen from the set $\{1000, 500, 300, 100\}$. Then we arbitrarily shut down an operational sensor every ρ time units. Going beyond that boundary is not possible, for nodes will then be failing at a much faster pace than they can be replaced and the algorithm will not behave well. Fortunately, real-world sensing units last much longer.

Two robots departing from $(0; 0)$ and $(500; 250)$ respectively try to optimize the coverage radius of a hostile environment of size 500×500 as much as they can. As before, the robot capacity is 5 sensors. This time, however, CBFCF will run for $5,000$ time units. Nodes fail at regular intervals of length ρ.

Displayed in Fig. 3.10(a) are the percentages of messages transmitted by sensors and robots, averaged over 50 independent runs, for each value of ρ and contrasted to those issued in normal conditions (i.e. no sensor failures, $\rho = \infty$). Figure 3.10(b) depicts the same information but in terms of M and V.

[3]Recall that V can also be interpreted as the number of *reply* messages from sensors missed by the robot.

Both charts unveil the negative impact of unreliable sensing units over the network stability. As sensors fail more often (i.e. smaller values of ρ), its still active neighbors have to emit *failure notification* messages and search agent nodes have to perform two rounds of face traversal along the network border to make sure everyone knows about it.

Figure 3.10(a) displays the superior growth of SM vs. RM as the failure probability increases. This is because CBFCF uses the alive sensors to maintain the integrity of the network topology via the chain of *search*, *reservation*, *deletion* and *cancelation* messages, which by large follow an elongated trajectory before discarded by a node in the outer face. Robots, on the other hand, will issue more *search* messages after detecting an empty spot, but their communication overhead is smaller compared to the sensors'.

Robots deplete their resources faster in unstable environments too, as shown by the steep incline of M and V in Fig. 3.10(b). The former is quite clear given that actuators now have to visit more empty TT vertices. The latter is less evident and follows this rationale: the fewer the active sensors, the lower the number of transmission paths that could be used to deliver the result of a *reply* message to a robot. Consequently, robot misses the *reply* and V becomes bigger.

The empirical test bed described in this section demonstrates that CBFCF can efficiently respond to the unexpected shutdown of sensing units under increasing severity levels. There must exist, however, an appropriate balance between robot capacity and failure rate to ensure full network repair. For scenarios where both parameters respect the tendencies observed in real-life settings, CBFCF exhibits a high degree of robustness.

3.8 Conclusions

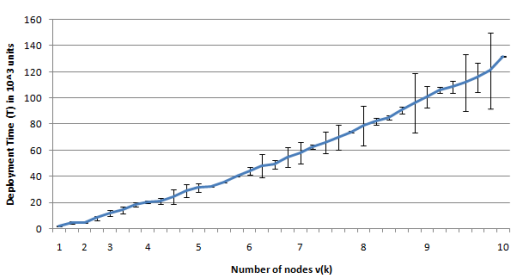
In this chapter, we studied for the first time the problem of carrier-based sensor deployment with reloading in the context of focused coverage. The proposed algorithm CBFCF is the first of its kind, to the best of our knowledge. All existing carrier-based sensor placement

protocols unrealistically require robots to carry all sensors at once (without reloading), thus ignoring their physical dimensions and the limited robot capacity. Furthermore, all of them target traditional coverage problems (e.g. "blanket"), as opposed to CBFCF which constructs a F-coverage formation, an emerging topology that is becoming increasingly important for critical surveillance operations.

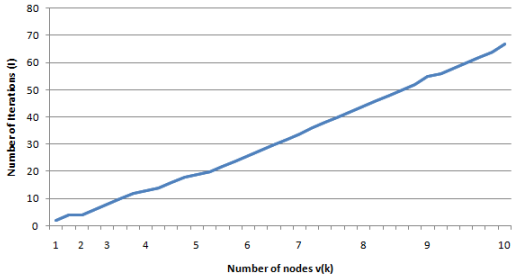
Several properties turn CBFCF into an appealing choice for its rapid incorporation to more challenging situations. It is strictly localized, obstacle-aware, efficient in terms of the inter-robot coordination mechanisms, guarantees maximal coverage in absence of sensor failures, remains robust when they occur, is cognizant about the robot cargo limitations and yields a bi-connected network layout around the POI.

One could think of extending our protocol to a complex environment where a series of continuous POIs exists, thus forming a Line of Interest (LOI). This could represent, for example, the trace of certain event or the boundary of a landmark (e.g. a building). Multiple discrete target objects, each modeled after a POI or a LOI, add more intricacy to the problem. The research goal would then be to derive an approach that preserves CBFCF's desirable features and attains some sort of shared coverage between nearby targets, possibly having irregular shapes, hence reducing the network size and eliminating harmful wireless interference.

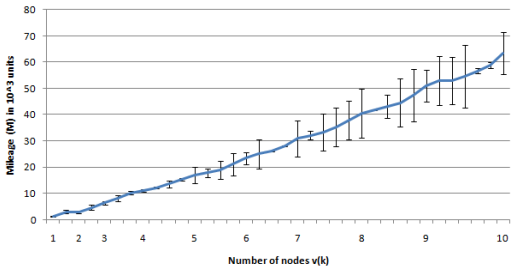
In the next chapter, we guide the reader throughout the second application scenario under consideration in which fault reactiveness in a WSN can be induced by mobile carrier robots.



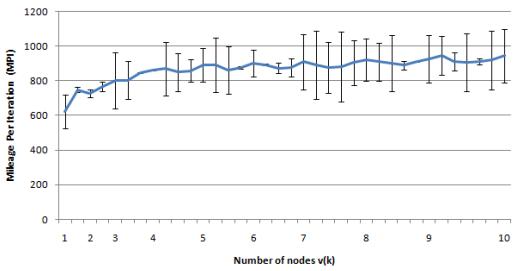
(a) Convergence time (T) in 10^3 units



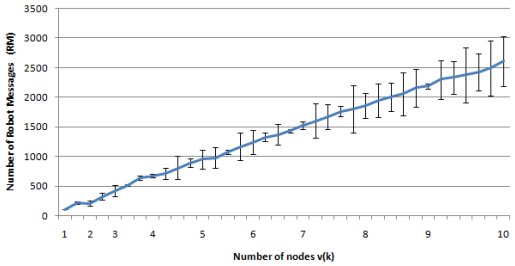
(b) Number of iterations (I)



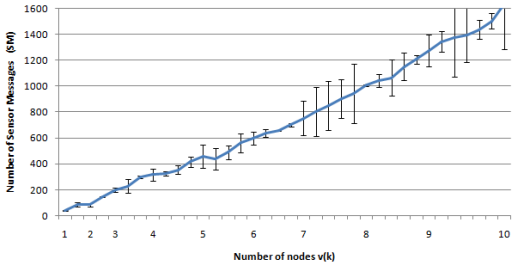
(c) Mileage (M) in 10^3 units



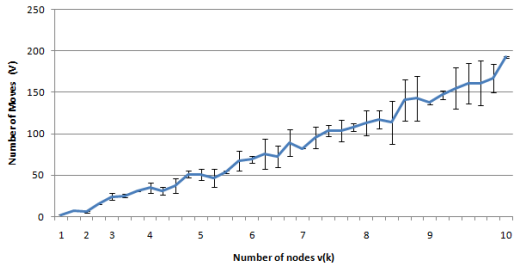
(d) Mileage per iteration (MPI)



(e) Number of robot messages (RM)

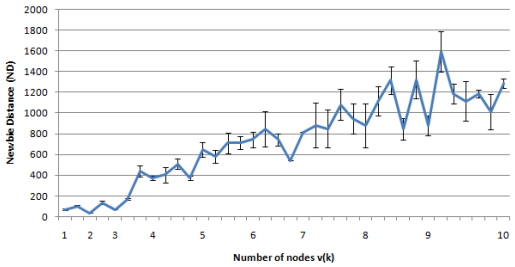


(f) Number of sensor messages (SM)

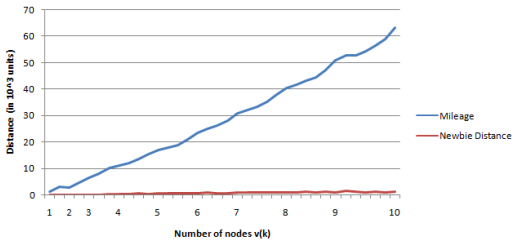


(g) Number of moves (V)

Figure 3.8: CBFCF's performance for varying network sizes in the same deployment field. The network size has been increased from $n = \nu(1) = 7$ to $n = \nu(10) = 331$.

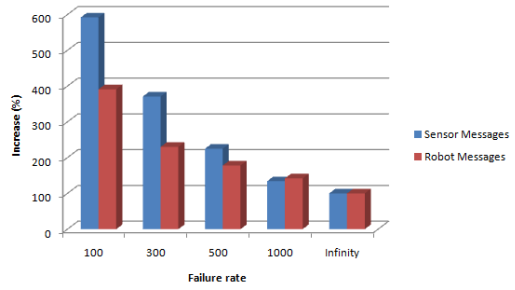


(a) The *newbie distance* as a function of the network size.

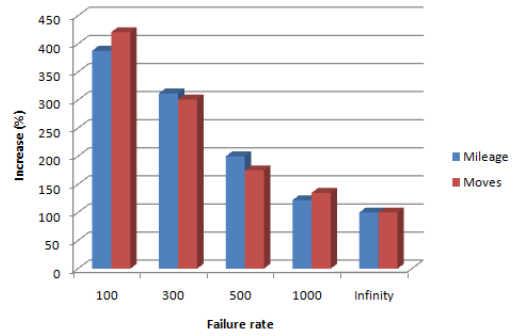


(b) Mileage versus newbie distance.

Figure 3.9: Illustration of the *newbie effect* in CBFCF.



(a) Percentage of increase in the transmitted messages



(b) Percentage of increase in the mileage and number of moves

Figure 3.10: CBFCF's performance at different sensor failure rates.

CHAPTER 4

SINGLE-CARRIER COVERAGE REPAIR

This chapter contains a thorough discussion on the *carrier-based coverage repair* (CBCR) problem, a real-life post-deployment setting arising in WSRNs that can be exploited to achieve fault reactiveness in the sensor-operated environment. After a short introduction to the general problem, the mathematical formulation of the solo-robot case (baptized as *single-carrier coverage repair*, SC^{2}R) is outlined. Then we present the first algorithm in literature that attempts to tackle SC^{2}R scenarios. It follows the algorithmic machinery encountered in artificial ant systems. Afterwards, a more advanced metaheuristic approach hybridizing ant colonies and genetic algorithms is unveiled. The latter scheme is able to provide significantly superior results compared to the former in identical time frames. A statistical methodology centered around a multilevel regression analysis is put forward. Its ensuing effect is the discovery of decision rules governing the activation of heuristic elements in the hybrid protocol to boost its performance when facing a particular problem instance.

4.1 Introduction

A quick glance at the CBCR problem was already presented in Sect. 1.2.2. In a nutshell, this is a sensor relocation scenario in a RA-WSN with two distinctive assumptions: (1) that

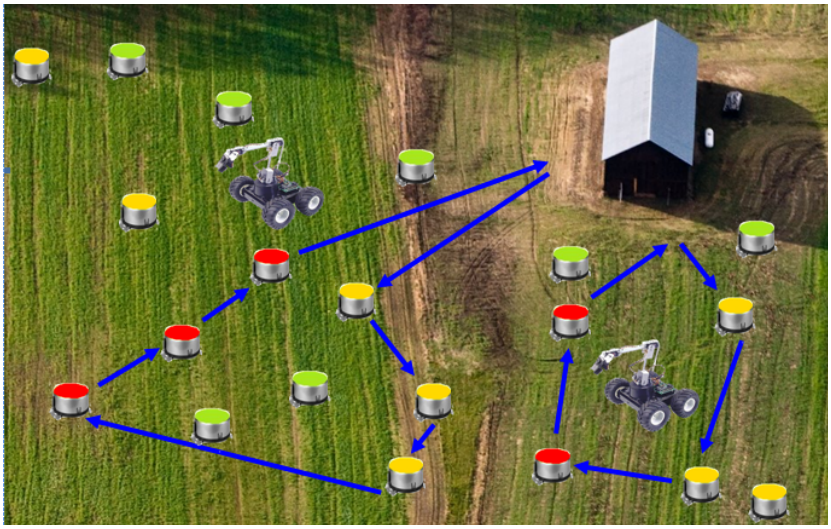


Figure 4.1: Illustration of the CBCR problem. Active, passive and damaged sensors are drawn in green, yellow and red, respectively. Blue arrows indicate the direction of the replacement trajectories.

the plethora of sensors scattered all over the field makes possible to partition the network into active and passive nodes by running an scheduling algorithm (e.g. [59]) so that the latter outnumber the former and (2) that a few mobile carrier robots are located at a base station, to which all network-wide information must flow (possibly via multi-hop transmission paths).

Upon detection of faulty (i.e. previously active) sensory units by neighboring nodes and the ensuing notification to the base station, an optimal route plan (set of disjoint, individual routes) for sensor replacement is to be computed at the centralized location, as portrayed in Fig. 4.1. By 'optimal' we mean a *feasible* route plan of corporate minimal cost. A route plan is considered feasible if each individual route is a Hamiltonian cycle originated at the base station, all damaged nodes are corporately replaced with passive nodes and the robot's capacity constraint is never violated along any route and no redundant pickups are collected. The *cost* of the route plan is defined here as the total distance traveled by all robots. Since the WSRN must quickly react to such faults, the time available for calculation of the robots' trajectories (routes) is usually short.

The particular case in which there is only one robot at the base station has been termed

as *single-carrier coverage repair*, (SC^{2}R) and this chapter is devoted to its study. For greater clarity, we will refer in this context to the single robot route as *robot tour*. The mathematical formalization of SC^{2}R as in integer linear programming (ILP) problem follows.

4.2 SC^{2}R Problem Formulation

In this section, the SC^{2}R problem is formally modeled as a unitary 1-TSP-SELPD instance. Any SC^{2}R scenario can be represented by a tuple $T = \langle G, \vec{q}, Q_0, Q \rangle$. A robot with initial cargo Q_0 and capacity Q has to pick up and deliver sensors in a network, represented by an optimization graph G and a demand vector $\vec{q}$.

The sensor network is described by a complete graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex set and E is the edge set, with e_{ij} being the directed edge from v_i to v_j. Each edge has an associated travel cost c_{ij}. Elements of V are either passive sensors (pickup customers) or damaged sensors (delivery customers). Each customer v_i has a unitary demand q_i (-1 for pickup customers and 1 for delivery customers) and $\vec{q} = (q_1, q_2, \ldots, q_n)$. Node v_1 represents the base station with demand $q_1 = 0$, V_D is the set of delivery customers and V_P is the set of pickup customers. The following statements hold.

$$V = \{v_1\} \cup V_D \cup V_P$$

$$V_D = \{v_i \in V : q_i = 1\}$$

$$V_P = \{v_i \in V : q_i = -1\}$$

$$V_D \cap V_P = \emptyset$$

A unique type of commodity (sensors) is to be transported by the robot from one place to another. The robot can carry at most Q sensors and leaves the base station with an initial cargo Q_0 ($0 \leq Q_0 \leq Q$). Not all graph nodes have to be visited. To ensure the service of all damaged sensors, a delivery node is assigned a profit p_i of 1, as is also done for the base station v_1. The

profit p_i of a pickup node is set to zero.

The goal is to find a minimal-cost feasible tour $\vec{\varphi}_{min}$. A tour $\vec{\varphi}$ is said to be feasible if it has no repeated nodes, repairs all damaged sensors without collecting redundant pickups, starts and ends at the base station and never violates the robot's capacity. More formally, for a given tuple T, the length of any feasible tour $\vec{\varphi}$ (including the base station) and its cost can be calculated as in (4.1) and (4.2), respectively.

$$|\vec{\varphi}| = |\{e_{ij} \in \vec{\varphi}\}| = 2 \cdot |V_D| - Q_0 + 1 \tag{4.1}$$

$$f(\vec{\varphi}) = \sum_{e_{ij} \in \vec{\varphi}} c_{ij} \tag{4.2}$$

with c_{ij} being the Euclidean distance between v_i and v_j.

Decision variables are defined as follows:

- $x_{ij} = 1$ if edge $e_{ij} \in \vec{\varphi}$, else 0
- $y_i = 1$ if node $v_i \in \vec{\varphi}$, else 0
- l_{ij} = load of the robot traveling along edge e_{ij}

This leads to the following ILP formulation:

$$Min \sum_{e_{ij} \in E} c_{ij} x_{ij}$$

subject to

$$\sum_{v_j \in V \setminus \{v_i\}} x_{ij} = y_i \qquad \forall v_i \in V \tag{4.3}$$

$$\sum_{v_i \in V \setminus \{v_j\}} x_{ij} = y_j \qquad \forall v_j \in V \tag{4.4}$$

$$\sum_{e_{ij}: v_i, v_j \in S} x_{ij} \leq \sum_{v_j \in S \setminus \{v_k\}} y_j \qquad \forall S \subset V \setminus \{v_1\}, v_k \in S \tag{4.5}$$

$$\sum_{v_j \in V \setminus \{v_1\}} l_{1j} = Q_0 \tag{4.6}$$

$$\sum_{v_j \in V \setminus \{v_i\}} l_{ji} - \sum_{v_j \in V \setminus \{v_i\}} l_{ij} = q_i y_i \qquad \forall v_i \in V \setminus \{v_1\} \tag{4.7}$$

$$0 \leq l_{ij} \leq Q x_{ij} \qquad \forall e_{ij} \in E \tag{4.8}$$

$$\sum_{v_i \in V} p_i y_i = p_{min} \tag{4.9}$$

$$l_{ij} \text{ is integer} \qquad \forall e_{ij} \in E \tag{4.10}$$

$$x_{ij} \in \{0, 1\} \qquad \forall e_{ij} \in E \tag{4.11}$$

$$y_i \in \{0, 1\} \qquad \forall v_i \in V \tag{4.12}$$

The objective function minimizes total costs, expressed in terms of total distance traveled by the robot to fix all damaged nodes. Constraints (4.3) and (4.4) ensure that each node is visited at most once. Constraints (4.5) eliminate subtours that do not involve the base station, as proposed by Feillet et al [54] for the TSP with profits. The robot leaves v_1 with an initial load Q_0 as stated in (4.6). Expressions (4.7) and (4.8) represent vehicle loading constraints. The difference in load of a robot entering and leaving a node must equal the demand of that node. A robot can never carry more than its maximum capacity. Constraint (4.9) ensures that all damaged units and the base station are visited. The collected profit in the tour is set to a minimum value p_{min}, equal to the number of delivery nodes $|V_D|$ plus node v_1. Since

only delivery customers and the base station have a profit of 1, these nodes are automatically included in a tour. Finally, constraints (4.10), (4.11) and (4.12) define the domain of the decision variables.

4.3 MMAS: An Ant Colony Approach to SC^{2}R

Given that solving SC^{2}R entails a non-polynomial computational complexity, exact algorithms can not cope with but small problem instances (about 40–50 nodes at most). Therefore, we turn to approximate optimization techniques that guarantee finding a suboptimal solution of acceptable quality in a limited execution time. Nature-inspired metaheuristics [150] are among the most powerful tools in this area, as previously discussed in Chapter 2.

Our first attempt to tackle SC^{2}R lingered around artificial ant colonies, particularly on the MMAS model described in Sect. 2.1.2. The rationale behind the selection of artificial ant colonies as the underlying optimization engine lies in: (1) the increasing empirical evidence [39] that ACO together with local search techniques is among the best-ranked algorithms for working out a wide array of difficult combinatorial optimization problems; (2) the incremental construction of the solution featured by the ant agents, which allows us to manage the problem constraints in a straightforward fashion [1] as opposed to building the entire tour and discarding it afterwards; (3) the latest research findings on the superiority of ACO parallel implementations over peer metaheuristics for TSP posed in [83] and on enhanced exploration strategies in [122], which are of great importance for scalability purposes.

The application of ACO to real-life problems is ruled by the following design guidelines:

(1) Represent the problem as a graph whose vertices are the problem states and whose edges are the transitions between the states. The graph will be "walked" by the ants to iteratively build the solutions.

(2) Define the meaning of the pheromone information τ.

(3) Define the meaning of the heuristic preference η (a.k.a. visibility function).

(4) Use the ants' local memory as well as the heuristic function to meet the problem's constraints, if possible.

(5) Choose the ACO model that best fits the problem at hand.

(6) Combine ACO with local search methods to improve the quality of the solutions found.

(7) Tune the algorithm parameters.

The graph representation for SC^{2}R was already outlined in Sect. 4.2. Guidelines 2 and 3 are heavily problem-dependent and have a crucial bearing on the algorithm's ultimate performance. When we are dealing with problems like 1-TSP-SELPD, where the order in which nodes are visited does matter, the pheromone trails are associated with the graph edges so as to reward good visiting sequences. The ACO heuristic function traditionally used for TSP is $\eta_{ij} = 1/c_{ij}$, but in 1-TSP-SELPD the desirability of e_{ij} has to take into account the current cargo of the search agent (ant) in order to avoid infeasible solutions. More elaborate heuristic functions for our problem are devised in Sect. 4.3.2.

The selection of the ACO model that best fits the problem at hand is also a crucial design step. We have no knowledge of any simple ant model being applied to 1-PDTSP thus far, but there is strong empirical evidence that MMAS outperformed AS and its improved versions when tested against a wide assortment of TSP instances [131]. This is due to the strong exploitation of the best solutions discovered and the effective space exploration mechanism enforced by the dynamic pheromone bounds, which make MMAS a very appealing choice for our ongoing research efforts.

Although still some improvements in the quality of the tours can be attained by adding local search procedures and tuning of the MMAS parameters (as stated in guidelines 6 and 7), for the sake of simplicity we will not incorporate any of these mechanisms into our initial algorithmic proposal for 1-TSP-SELPD. They will prove, however, crucial building blocks of the hybrid version in Sect. 4.6.

4.3.1 Initialization

At the beginning of every iteration, ants must be initialized on different graph nodes, from which they will start adding components to their tours. This step is done usually at random for ACO models addressing the TSP. Yet in our problem, we must place an ant at a feasible node depending on its initial cargo Q_0. The feasible initial neighborhood N_k for the k-th ant, from which a node will be randomly chosen as its starting point, is defined as

$$N_k = \begin{cases} V_P, & \text{if } Q_0 = 0 \\ V_D, & \text{if } Q_0 = Q_{max} \\ V - \{v_1\}, & \text{otherwise} , \end{cases}$$

where $Q_{max} = \min(n_D, Q)$ and n_D is the number of remaining damaged nodes in the deployment region. Initially, $n_D = |V_D|$.

4.3.2 Heuristic Rules

The heuristic component in any ant algorithm describes the unchanging, problem-specific knowledge, as opposed to the pheromone trails which convey the dynamic, collective knowledge achieved by means of self-organization. In 1-TSP-SELPD, the heuristic desirability η_{ij} of transitioning from the current problem state v_i to any neighbor state v_j is posed as a function of their distance, the type of the current node (pickup or delivery) and the current cargo of the search agent. Several heuristic functions are envisioned and presented next, each trying to capture a desirable facet of the overall tour construction.

"DROP-OFF FIRST" RULE

If robot's cargo is empty, then choose the closest passive sensor. Otherwise, prioritize the drop-off operation by selecting the nearest damaged sensor as in (4.13).

$$\eta_{ij} = \begin{cases} \frac{\text{sgn}(Q_k)}{d_{ij}}, & v_j \in V_D \\ \frac{1 - \text{sgn}(Q_k)}{d_{ij}}, & v_j \in V_P \end{cases} \tag{4.13}$$

where sgn(·) is the sign function, Q_k the current cargo of the k-th ant and d_{ij} the Euclidean distance between v_i and v_j.

"PICKUP FIRST" RULE

If the robot is full, drop a sensor at the closest delivery. Otherwise, prioritize the pickup operation by choosing the nearest passive sensor as in (4.14).

$$\eta_{ij} = \begin{cases} \frac{\left\lfloor \frac{Q_k}{Q_{max}} \right\rfloor}{d_{ij}}, & v_j \in V_D \\ \frac{1 - \left\lfloor \frac{Q_k}{Q_{max}} \right\rfloor}{d_{ij}}, & v_j \in V_P \end{cases} \tag{4.14}$$

where $\lfloor \cdot \rfloor$ is the floor function. Notice that $Q_k \leq Q_{max}$, i.e., the current cargo must never exceed the number of defective nodes left in the terrain (extra load) or the maximum cargo capacity.

"NEAREST NEIGHBOR" RULE

Whenever both drop-off and pickup are possible, always choose the nearest spot according to (4.15).

$$\eta_{ij} = \begin{cases} \frac{\left\lceil \frac{Q_k}{Q_{max}} \right\rceil}{d_{ij}}, & v_j \in V_D \\ \frac{1 - \left\lfloor \frac{Q_k}{Q_{max}} \right\rfloor}{d_{ij}}, & v_j \in V_P \end{cases} \tag{4.15}$$

where $\lceil \cdot \rceil$ is the ceiling function.

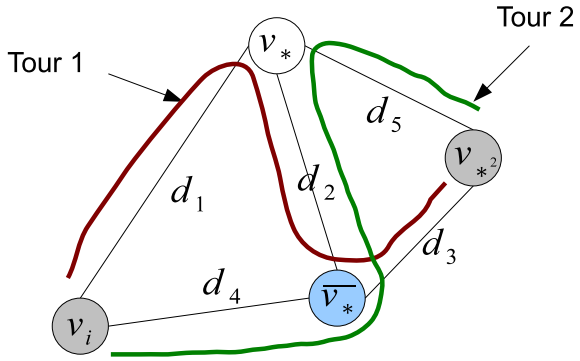


Figure 4.2: The LOOK AHEAD heuristic rule

"LOOK AHEAD" RULE

This is an extension of the previous rule but from a rather futurist perspective. Let v_i be the current node and v_D and v_P the nearest damaged and passive sensor to v_i, respectively. Let us denote by v_* the location (either pickup or drop-off) that the robot will go to by the NEAREST NEIGHBOR (NN) rule. Let $\overline{v_*}$ be v_P if $v_* = v_D$ or v_D otherwise. Finally, denote by v_{*^2} the location that the robot will advance to, by the NN rule, after it arrives at v_*, as in Fig. 4.2.

We are now ready to enunciate the look-ahead rule as follows. If $v_{*^2} = \overline{v_*}$, the ant will simply follow the NN rule for next node selection. Otherwise, it will compute two tours for v_*, $\overline{v_*}$ and v_{*^2}: $tour_1 = v_i \rightarrow v_* \rightarrow \overline{v_*} \rightarrow v_{*^2}$ and $tour_2 = v_i \rightarrow \overline{v_*} \rightarrow v_* \rightarrow v_{*^2}$, with different next node but the same destination v_{*^2}. The ant will select the node that leads to a shorter tour. In case of tie, preference is given to the delivery customer.

The analytical expression corresponding to the look-ahead rule is given below:

$$\eta_{ij} = \max\left(0, q_j^{\operatorname{sgn}\left(\left\lfloor \frac{d_5}{d_1+d_3-d_4} \right\rfloor\right)} \times -\frac{q_i}{d_{ij}}\right), \tag{4.16}$$

where $d_1 = d(v_i, v_*)$, $d_3 = d(\overline{v_*}, v_{*^2})$, $d_4 = d(v_i, \overline{v_*})$ and $d_5 = d(v_*, v_{*^2})$.

MAJORITY RULE

Choose as next node the most voted one among the outcomes provided by all previous heuristic rules. Arbitrarily break ties.

STOCHASTIC RULE

Randomly select any of the previous rules at each step of the tour construction.

Notice that all heuristic rules evaluate to zero for those nodes that will make the tour infeasible due to violation of the capacity constraint. Subsequently, the transition probability computed in (2.1) for such nodes will also be zero, thus pruning them from the feasible neighborhood N_k from which the next component emerges after stochastic roulette wheel selection.

4.3.3 Pheromone Update

The pheromone values in our ACO-based system are updated as in (2.2) and $\Delta\tau_{ij}$ is computed as shown below:

$$\Delta\tau_{ij} = \begin{cases} \frac{R}{f(\vec{\varphi}_{ib})}, & \text{if } e_{ij} \in \vec{\varphi}_{ib} \\ 0, & \text{otherwise} , \end{cases} \tag{4.17}$$

where $\vec{\varphi}_{ib}$ is the best tour produced in the current iteration, $f(\vec{\varphi}_{ib})$ its associated cost, R is called *"reward factor"* and set to the cost of the best solution found in the first iteration. The shorter the tour computed in subsequent iterations, the larger the amount of pheromone added to its edges. This update rule helps drive the search towards good solutions.

The pheromone bounds are updated as follows:

$$\tau_{max} = \frac{1}{\rho} \cdot \frac{R}{f_(\vec{\varphi}_{opt})} , \tag{4.18}$$

$$\tau_{min} = \frac{\tau_{max} \cdot (1 - (p_{best})^{1/|\vec{\varphi}_{opt}|})}{J_{avg} \cdot (p_{best})^{1/|\vec{\varphi}_{opt}|}} , \tag{4.19}$$

where $\vec{\varphi}_{opt}$ is the best-ever-found solution, p_{best} stands for the probability that the best solution is constructed again and J_{avg} is the average number of options available at the graph nodes.

4.3.4 Stop Criterion

The algorithm terminates when an upper bound of the number of iterations is reached. In this way, the WSN can promptly respond to the presence of faulty sensors in the field.

4.4 Addressing Scalability Concerns

As the size of the monitoring area grows, so it does the number of sensors required for its surveillance. This means that the increased dimensionality of the SC^2R instances might strike the algorithm's performance unless it scales well to large systems. Scalability is a highly coveted property in optimization methods, often sought through paralellism [83] given the intrinsically distributed nature shared by all population-based metaheuristic algorithms.

4.4.1 Two-Step Ant Colony Optimization (TS-ACO)

Acceleration can also be accomplished if we approach the initial search state to the final (goal) state as much as possible. This is the chief thought behind the novel "two-step" exploration strategy for ant algorithms (TS-ACO) [121, 122]. Briefly described, the search process undertaken by the ants is split into two stages. In the first stage, a subset of the ants in the entire colony walk throughout the graph in a subset of the total iterations trying to find optimal subtours, i.e., tours with lower dimensionality than the actual tours. The results of the exploration are actively exploited to speed up the algorithm's convergence, either by utilizing the best subtours found as initial subsolutions for the remaining ants in the second stage or by reusing the corporate knowledge expressed in the pheromone matrix computed during the first stage. The former version is referred to as "solution-driven" (S-TS-ACO) whereas the latter was baptized as "pheromone-driven" (P-TS-ACO).

The split factor $0 < r < 1$ is a TS-ACO parameter that controls the allocation of resources to each search phase. For instance, $r = 0.45$ means that, during the first stage, 45% of the individuals in the colony will look for subtours of length 45% of the desired tours in just 45% of the total number of cycles (iterations). In stage 2, the remaining ants will run for the leftover number of cycles trying to find solutions to the original problem.

In S-TS-ACO, the best ns subtours of the first search phase were recorded. Now, each ant in the second phase is randomly initialized on a graph node from which it will resume the search in order to complete its tour, which was randomly picked from one of the best ns partial solutions. The pheromone matrix is reinitialized. However in the P-TS-ACO version, the ants in the second phase start their search from scratch but lean on the pheromone matrix computed during the previous phase.

Extensive empirical and theoretical studies developed in [121] with TSP and the Quadratic Assignment Problem (QAP) concluded that TS-ACO maintains the quality of the solutions found by ACO yet shortens the computational time by $40\% \sim 60\%$. We take advantage of this finding to implant scalability in our two-step MMAS scheme (TS-MMAS).

4.4.2 Applying TS-MMAS

Recall from (4.1) that the tour dimensionality in SC^2R is a function of the number of damaged nodes reported at the base station. Hence, we may split the search by following either of two criteria: the number of visited nodes (tour length, denoted as TS-TL) or the number of visited damaged nodes (TS-DN). Each criterion may prove useful under different topological circumstances.

Being m the colony size, nc the maximum number of cycles and having decided on the values of r and ns, we allocate the resources for the two search stages as follows:

- Number of ants: $m_1 = \lfloor r \cdot m \rfloor$ and $m_2 = m - m_1$
- Number of cycles: $nc_1 = \lfloor r \cdot nc \rfloor$ and $nc_2 = nc - nc_1$

- Solution dimensionality: In the first stage under TS-TL, find subtours of length $tl_1 = \lfloor r \cdot |\vec{\varphi}| \rfloor$. For TS-DN, find subtours that have visited exactly $tl_1 = \lfloor r \cdot |V_D| \rfloor$ defective units. In the second stage and irrespective of the split criterion, find tours of length $tl_2 = |\vec{\varphi}|$.

Here m_1, nc_1 and tl_1 are the resources allocated for the first stage while m_2, nc_2 and tl_2 are those of the second stage.

4.5 MMAS Experimental Study

We have conducted a set of experiments to test the feasibility of our initial solution approach for SC^{2}R. Simulations were run in MATLAB with an AMD Athlon 64 X2 Dual Core at 2.6 GHz and 2 GB of memory under Windows Vista. Our benchmark instances[1] are very similar to those in [71]. For each value of n, the number of nodes in the graph, random 2D coordinates for $n-1$ customers were generated in the square $[-500, 500]^2$. The difference lies in that all demands are unitary ones $\{-1, 1\}$ and the ratio of damaged to passive sensors was set to 1:3. The depot is placed at $(0, 0)$ and has demand $q_1 = 0$. The travel cost d_{ij} was computed as the Euclidean distance between v_i and v_j. Each instance file is characterized by the number of sensors n and the robot capacity Q and can be split into two classes: *small* instances with $n \in \{20, 30, 40, 50, 60\}$ and *large* instances with $n \in \{100, 200, 300, 400, 500\}$. As to the robot capacity Q, we have considered three possible values: $Q \in \{25\% \cdot |V_D|, 50\% \cdot |V_D|, 100\% \cdot |V_D|\}$ given that the robot departs from the base station with no extra load.

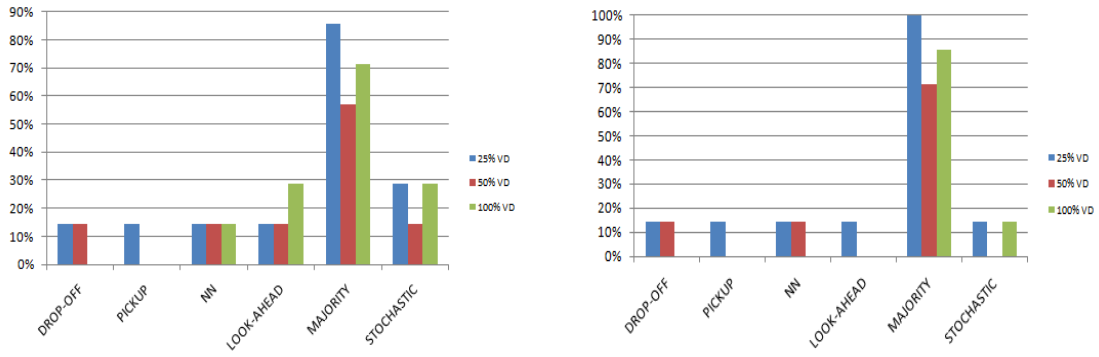
4.5.1 Experiment 1: The ACO Heuristic Rules

Traditionally, ACO researchers have focused on improving the collective behavior (i.e., pheromone trails) of the search agents in an attempt to reach good suboptimal solutions. Somehow, the problem-specific knowledge (heuristic component) has been neglected. In our first

[1] Available at `http://www.site.uottawa.ca/~rfalc032/files/MMAS_2010_data.zip`

Table 4.1: Parameters of the MMAS approach

Name	Description
colony_size	Number of ants in the colony at every cycle.
α	Sensitivity coefficient of the pheromone trails.
β	Sensitivity coefficient of the heuristic information.
ρ	Pheromone persistence rate.
p_{best}	Probability that the best solution is constructed again.



(a) % of tested files where the best **overall** solution was found per heuristic rule

(b) % of tested files where the best **average** solution was found per heuristic rule

Figure 4.3: Performance of the six heuristic rules in the MMAS model

experiment, we are to assess the bearing of the six heuristic rules portrayed in Sec. 4.3.2 on the overall performance of the proposed scheme. To this end, we have run MMAS for 1,000 iterations over our benchmark instances by using one heuristic rule at a time. Table 4.1 lists all MMAS parameters . Their values have been set after a thorough empirical examination as follows: $\alpha = 2, \beta = 3, \rho = 0.9, p_{best} = 0.1$ and the colony size equals 15 for small instances and 30 for large instances. The best and average tour costs over 10 independent executions per heuristic rule are reported.

Figures 4.3(a) and 4.3(b) reveal an interesting finding: the MAJORITY rule outperforms every other heuristic function regardless of the robot capacity or network density (number of nodes in the graph), in terms of the best overall and average solutions ever found. In the latter case, MAJORITY always reports the best average solution among peers. This fact sheds light on the need of reaching a consensus among several election criteria at the node level. The

holistic view of the ant's neighborhood seems to be much more promising than its assessment from biased standpoints. Yet the computational time of the MAJORITY rule is the longest one and we might not always afford such delay in the calculation of the robot trajectory, especially when immediate coverage repair is demanded. Fortunately, in such cases we may opt to apply the STOCHASTIC rule instead. Although far off the encouraging performance of MAJORITY, a stepwise randomization in the heuristic decision-making process has proved more competitive than conventional rules such as NN. The occasional application of MAJORITY within the stochastic selection yields profitable results.

Overall, LOOK AHEAD is more effective than heuristic rules 1 – 3 in Sec. 4.3.2, with a better demeanor in scenarios where the carrier capacity is not overly restricted. Another observation is that always prioritizing some activity (either drop-off or pickup) does not lead to good results, as displayed in the previous charts. This is also consistent with the finding in [72] that joining customers of different type increases the probability of finding a good solution for 1-PDTSP, which is a particular case of 1-TSP-SELPD.

4.5.2 Experiment 2: Performance Comparison

Despite our ACO-based method being the first known algorithm that targets SC^2R and thus having no competitor in literature, we can still contrast its performance with that of a recently enhanced version of Simulated Annealing named IMSA [100] because no customized configuration is required for IMSA. Other published approaches such as [70] were not considered for benchmarking purposes given that they are tailored to attack a different problem (1-PDTSP), hence there is no common ground for a fair comparison baseline.

For the sake of brevity, we will only outline the most restricted (and challenging) scenario in which the robot capacity Q is only a quarter of the total number of damaged nodes. Tables 4.2 and 4.3 report the best cost, average cost and average execution time over 10 independent runs for IMSA and MMAS, respectively. Their standard deviations (in %) are included too.

Highlighted in boldface are the best tour costs per instance file, for which meaningful

Table 4.2: IMSA performance with $Q = 25\% \cdot |V_D|$

n	Q	Best cost	Avg cost	Std dev. cost (%)	Avg time	Std dev. time (%)
20	1	**2579.59**	2785.23	5.59	20.62	7.86
30	2	3572.71	3842.03	4.38	25.07	6.78
40	3	4343.67	4752.25	6.43	26.77	1.42
50	3	5211.06	5747.04	5.23	33.68	8.97
60	4	6432.87	7178.91	6.22	34.43	2.03
100	6	10629.87	11637.17	5.05	98.02	0.40
200	13	23125.5	24302.86	4.25	160.89	4.92
300	19	37865.17	40627.04	4.45	222.16	5.26
400	25	54204.18	56362.71	3.33	331.51	0.09
500	31	65559.9	70719.13	3.57	397.35	4.40

differences between the algorithms were uncovered by a non-parametric Iman-Davenport test running at a significance level of 5%, thus supporting the superiority of MMAS versus IMSA. However, the same test also detected a significant gap in the running time of each optimization scheme, this time in favor of IMSA. Such finding makes sense in light of the fact that IMSA performs no maintenance (update) of a global communication structure like the pheromone matrix found in all ACO algorithms, for which some computation time has to be set aside. Additionally, MMAS maintains a set of concurrent solutions (ants) per iteration, whereas IMSA is a trajectory-based algorithm.

4.5.3 Experiment 3: The Two-Step Exploration Strategy

With the aim of accelerating convergence, we have implemented the four variations of TS-MMAS previously outlined in Sect. 4.4. Several values for the split factor r were tried: 0.2, 0.25, 0.3 and 0.35; the empirical evidence supported $r = 0.3$ as the most profitable one. Parameter ns was set to 5% of the number of subsolutions calculated in the first stage, a rule of thumb enunciated in [121] after a thorough performance analysis.

All along, it seems that the pheromone-driven exploration scheme (P-TS-MMAS) outperforms the solution-driven version (S-TS-MMAS) for 75% of the data sets under consideration.

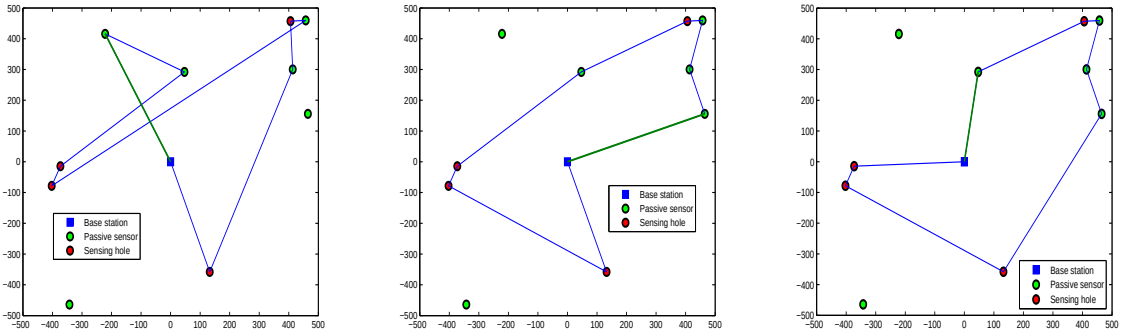
Table 4.3: MMAS performance with LOOK-AHEAD and $Q = 25\% \cdot |V_D|$

n	Q	Best cost	Avg cost	Std dev. cost (%)	Avg time	Std dev. time (%)
20	1	**2579.59**	2579.59	0	37.99	4.2
30	2	**3273.18**	3416.31	0.45	58.96	5.23
40	3	**3460.33**	3593.92	1.92	89.41	3.74
50	3	**3933.39**	4093.7	0.85	109.12	2.06
60	4	**3741.19**	4053.19	3.58	137.87	0.17
100	6	**6288.55**	6619.84	0.94	374.8	4.01
200	13	**10748.05**	11277.01	2.82	942.25	3.67
300	19	**15852.06**	21705.81	5.22	457.6	2.91
400	25	**27681**	30951.95	4.25	551.08	2.13
500	31	**40812.4**	45562.97	4.2	614.05	3.17

In other words, the collaborative knowledge embedded in the pheromone matrix after the construction of the initial subtours seems to be a more effective means for driving the ants to good regions of the discrete search space. This observation is statistically supported by an Iman-Davenport test that uncovered significant differences among the four two-step variants and then by a corrected Bonferroni-Dunn test with P-TS-MMAS as the control algorithm.

Of the two criteria used to define the subsolution dimensionality (i.e., number of visited defective nodes vs. tour length), the latter yields more encouraging outcomes across the majority (80%) of the data archives tested. The rationale behind this claim has much to do with the topological features of the network under consideration. For instance, in Fig. 4.4(a) we illustrate the undesirable effect of building tours that visit a subset of the damaged sensors under the solution-driven exploration model, for these subtours might be far from optimality yet they will serve as starting solutions for the ants in the second stage, thus likely leading to long robot trajectories. In general, this situation can be relieved by changing the dimension of the subsolutions to be found and let them be a function of the total tour length, as in Fig. 4.4(b). Next, in Fig. 4.4(c) we show how the pheromone-driven model was able to find the optimal robot trajectory.

In a nutshell, the application of TS-MMAS in SC^2R settings produced high-quality so-



(a) A tour by S-TS-MMAS-DN. Note the effect of using as starting solution for stage 2 an "optimal" subtour that visits exactly two damaged sensors.

(b) A tour by S-TS-MMAS-TL. Initial subsolutions for the second stage are those feasible tours that visit exactly two nodes (of any type).

(c) P-TS-MMAS-DN found the optimal network solution. Ants in stage 2 build their solution from scratch using the pheromone matrix in stage 1.

Figure 4.4: Robot trajectory computed with $r = 0.3$, $Q_0 = 0$ and $Q = 3$. The green line symbolizes the departing direction.

lutions (yet not superior in most cases to those found by standard MMAS, as demonstrated by the Iman-Davenport test which yielded no significant differences between them) but in a time frame equivalent to $35\% \sim 70\%$ of MMAS' running time, greatly contributing to the robustness of the algorithm when dealing with densely deployed networks.

As encouraging as these preliminary results are, further performance gains could still be obtained if we improve the quality of the initial population and foster greater diversification during the search process. The intended goal is successfully accomplished in the following section through the blending of ACO with genetic algorithms.

4.6 GA-ACO: The Hybrid Metaheuristic Approach

In this section a GA-ACO approach to tackle SC^2R settings is proposed. The main idea is to combine GA's exploratory power with an intensive exploitation of good visiting sequences based on ACO's pheromone trails. The pheromone-based crossover operator recently developed in [154] has been adopted, but now applied to a different problem (1-TSP-SELPD), which requires an extra layer of intricacy, i.e. finding the right set of nodes in the tour. Thus, a higher

degree of diversification is provided in two complementary manners: (1) via problem-specific local search operators and (2) via an adaptive mechanism that mutates parent individuals before crossover.

Cyclic path representation was adopted as the solution encoding. Every chromosome encodes a tour $\vec{\varphi}$ as a vector of $|\vec{\varphi}|$ dimensions. The tour starts at v_1's location and visits nodes to the right in a closed loop until it finally ends at v_1. For example, $\vec{\varphi}$ = (5, 3, 8, 1, 7, 6, 2, 4) means the robot departs from v_1, then goes to v_7, v_6, v_2, v_4, v_5, v_3 and v_8 before returning to v_1.

Foundational aspects of GA and ACO were previously discussed in Sect. 2.1.2 and 2.1.2, respectively. An overview of our algorithm is first presented in subsection 4.6.1, then its building blocks are discussed in more detail in subsections 4.6.2 to 4.6.8.

4.6.1 Algorithmic Overview

Algorithm 4 displays the workflow of GA-ACO. First, our approach builds a strong initial population by means of an efficient construction heuristic *GenerateIndividual( )* and local search operators *ImproveIndividual( )*. A preprocessing step in lines 1 and 2 generates two lists of customers, which are used in these two routines. In line 7, the *EvaluatePopulation( )* function calculates the fitness value (tour cost) of each chromosome in the initial population and keeps track of the best-performing individual $\vec{\varphi}_*$. Next, this initial population is used in the GA-ACO algorithm in lines 8 to 23. Two individuals are randomly selected for crossover and probabilistically mutated according to their age in the population. *SelectCrossoverParents( )* picks two chromosomes through fitness-based roulette wheel selection [81] to perform as parents in the upcoming crossover. A random number uniformly distributed in [0,1] is generated by *rand( )* in line 12 to decide whether pheromone-based crossover will be carried out with a given probability p_{CR}. Before crossover, selected parents may first be mutated by the *ApplyMutation( )* routine in lines 13 and 14. The function *ApplyCrossover( )* is responsible for performing the crossover operation in line 15, after which the offspring is further improved via local search in line 16. At most half of the individuals in the population are replaced with

newly generated offspring. *UpdatePopulation( )* substitutes low-quality population members with the offspring created after every generation. Fitness values are recalculated and $\vec{\varphi}_*$ updated in line 21. The *UpdatePheromoneTrails( )* routine in line 22 proceeds to update the pheromone matrix and a new generation arises until the termination criterion is met. Detailed steps of the GA-ACO algorithm are outlined in the following subsections.

Algorithm 4 Hybrid GA-ACO for SC^2R problems

Input: pop_size, cl, p_{CR}, β, p_0, ageIni, maturity, p, ρ
Output: Best ever-found individual $\vec{\varphi}_*$

1: CL ← *ComputeCandidateList( )*;
2: CPL ← *ComputeCandidatePickupList( )*;
3: **for** i = 1 to pop_size **do**
4: $\vec{\varphi}_i \leftarrow$ *GenerateIndividual*(CPL);
5: $\vec{\varphi}_i \leftarrow$ *ImproveIndividual*($\vec{\varphi}_i$, CL);
6: **end for**
7: *EvaluatePopulation( )*;
8: **while** termination criterion not met **do**
9: *offspring.clear( )*;
10: **for** i = 1 to pop_size / 2 **do**
11: $[\vec{\varphi}_{p_1}, \vec{\varphi}_{p_2}] \leftarrow$ *SelectCrossoverParents( )*;
12: **if** *rand( )* $\leq p_{CR}$ **then**
13: $\vec{\varphi}_{p_1} \leftarrow$ *ApplyMutation*($\vec{\varphi}_{p_1}$) with $p_M(\vec{\varphi}_{p_1})$;
14: $\vec{\varphi}_{p_2} \leftarrow$ *ApplyMutation*($\vec{\varphi}_{p_2}$) with $p_M(\vec{\varphi}_{p_2})$;
15: $\vec{\varphi}_{child} \leftarrow$ *ApplyCrossover*($\vec{\varphi}_{p_1}, \vec{\varphi}_{p_2}$);
16: $\vec{\varphi}_{child} \leftarrow$ *ImproveIndividual*($\vec{\varphi}_{child}$, CL);
17: *offspring.add*($\vec{\varphi}_{child}$);
18: **end if**
19: **end for**
20: *UpdatePopulation( )*;
21: *EvaluatePopulation( )*;
22: *UpdatePheromoneTrails( )*;
23: **end while**
24: **return** best-ever individual $\vec{\varphi}_*$;

4.6.2 Parameters

Table 4.4 lists all parameters involved in GA-ACO's configuration. In the following subsections, the role of these parameters is discussed in detail. Their proper values are analyzed in Section 4.7. The value of the parameter p_M (describing the probability of parent mutation)

Table 4.4: Parameters of the GA-ACO approach

Name	Description
pop_size	Number of individuals in the population at every generation.
cl	Size of the candidate pickup list and candidate list.
p_{NFC}	Probability of selecting the nearest feasible customer during population initialization.
p_{CR}	Crossover probability.
β	Sensitivity coefficient of the heuristic information in ACO.
p_0	Probability of "greedy" next element selection during ACO-based crossover.
ageIni	Number of generations before parent mutation occurs.
maturity	Aging threshold for a chromosome.
p	Percentage of pickup nodes to be replaced during mutation.
ρ	Pheromone persistence rate.

is defined by other parameters (subsection 4.6.6). Therefore, p_M is neither listed as input in Algorithm 4 nor in Table 4.4.

4.6.3 Population Initialization

An initial population of pop_size individuals (chromosomes) is generated. Good initial solutions may speed up convergence to the global optimum. In our algorithm, the population is initialized with feasible chromosomes of acceptable quality. This process is undertaken in two steps. First, we choose a subset of the problem nodes (graph vertices) that guarantee coverage repair. Then, a feasible node permutation is sought. The *GenerateIndividual( )* procedure in line 4 proceeds as follows:

Step 1. Guarantee coverage repair.

- This step relies on the Candidate Pickup List (CPL), a static structure computed in line 2 and holding the 'cl' nearest pickups to every delivery customer. It may contain less than 'cl' entries if there are not enough pickups.

- Generate an initial vector $\vec{vec}$ with dimension as in (4.1) containing all damaged sensors and the base station. Randomly select exactly $|V_D| - Q_0$ deliveries (i.e. elements of V_D)

and arbitrarily draw a pickup out of the CPL of each chosen delivery. Make sure there are no repeated pickups and complete $\vec{vec}$ with those elements.

- Shuffle elements in $\vec{vec}$.

Step 2. Find a feasible permutation.

- Represent a tour $\vec{\varphi}$ by a vector of dimension as in (4.1). Select a random index $k \in \{1..|\vec{\varphi}|\}$ for the base station and set $\vec{\varphi}(k) = v_1; l = Q_0$; where l is the current load of the robot.
- $next \leftarrow$ the nearest feasible customer to $\vec{\varphi}(k)$ in $\vec{vec}$ (i.e. the element in $\vec{vec}$ that causes no capacity violation when inserted cyclically after $\vec{\varphi}(k)$ and is the closest in distance to it) with probability p_{NFC} or a random, feasible one otherwise.
- Move to next index $k = \text{mod}(k, |\vec{\varphi}|) + 1$; set $\vec{\varphi}(k) \leftarrow next$; and update the robot's cargo $l \leftarrow l - q_{next}$;
- Repeat the two previous operations until all elements in $\vec{\varphi}$ are allocated.

Notice that Step 1 is quite stochastic (fostering population diversity) while Step 2 is fairly deterministic (aiming at finding the shortest tour) in an attempt to balance exploration and good fitness values during the conception of the initial population.

4.6.4 Improving Candidate Solutions

The *ImproveIndividual( )* procedure strengthens the quality of the individuals in the initial population and of any ensuing offspring through the application of general and problem-specific local search operators in the order listed below. Each induced neighborhood is only partially explored, i.e. until the first improving solution is encountered, so that this phase does not become very time-consuming.

The Or-OPT Operator

This operator introduced in [112] aims at changing the layout of the current tour $\vec{\varphi}$ by inserting every tour node into all of the $|\vec{\varphi}| - 1$ untried positions.

The Pickup Exchange (PEX) Operator

This problem-specific operator does not alter the tour layout, but replaces each pickup node $v_p \in \vec{\varphi}$ with the first improving pickup $v_{p'} \notin \vec{\varphi}$ that appears in the CPL of any adjacent damaged sensor v_{d_1} or v_{d_2}, as depicted in Algorithm 5. Experimental evidence in Sect. 4.7 will confirm that this pivotal exploration step has a significant bearing upon the performance of the proposed metaheuristic method.

Algorithm 5 PEX Operator for SC^{2}R instances

Input: tour $\vec{\varphi}$
Output: improved tour $\vec{\varphi}'$
1: $\vec{\varphi}' \leftarrow \vec{\varphi}$; $L \leftarrow$ the set of pickup nodes in $\vec{\varphi}$;
2: **for** each pickup $v_p \in L$ **do**
3: **if** $\exists$ a neighboring delivery v_{d_1} or v_{d_2} in $\vec{\varphi}'$ **then**
4: Find a still unvisited $v_{p'} \in \text{CPL}(v_{d_1}) \cup \text{CPL}(v_{d_2})$;
5: $\vec{\varphi}'' \leftarrow$ replace v_p with $v_{p'}$;
6: $\vec{\varphi}' \leftarrow$ the first feasible tour $\vec{\varphi}''$ that improves $\vec{\varphi}'$;
7: **end if**
8: **end for**

The 2-OPT Operator

2-OPT [39] modifies the tour layout whenever a promising exchange of any two pairs of edges is found. This process is repeated until no more improvements are possible. In order to shorten the running time, we solely consider edges e_{ij} to be inserted such that $j \in \text{CL}(i)$, where $\text{CL}(i)$ is the candidate list of node v_i, a data structure computed in line 1 and holding the 'cl' closest customers to v_i. Notice that this is not the CPL in Section 4.6.3, which only contains pickup nodes.

4.6.5 Pheromone-Based Informed Crossover

The idea in [154] of embedding ACO pheromone trails into an informed crossover operator is exploited here, with the purpose of reusing good visiting sequences previously learned via stigmergy (indirect communication) of the artificial ants. When two parents are selected, a crossover is executed with probability p_{CR}. The *ApplyCrossover( )* method relies on parent chromosomes $\vec{\varphi}_{p_1}$ and $\vec{\varphi}_{p_2}$ and behaves as follows:

- Initialize a vector $\vec{\varphi}_{child}$ of size as in (4.1) and randomly generate $k \in \{1..|\vec{\varphi}_{child}|\}$. Set $\vec{\varphi}_{child}(k) \leftarrow v_1; l \leftarrow Q_0$; where l is the current robot load.
- $N \leftarrow$ set of feasible "neighbors" (i.e., adjacent cyclic entries) of $\vec{\varphi}_{child}(k)$ in $\vec{\varphi}_{p_1}$ and $\vec{\varphi}_{p_2}$.
- If $N \neq \emptyset, next \leftarrow$ nearest element in N to $\vec{\varphi}_{child}(k)$.
- Else, $F \leftarrow$ the set of all feasible next nodes in the problem. Select *next* after the pseudo-random probabilistic rule in (4.20), where $i = \vec{\varphi}_{child}(k)$, $\tau_{ij}(t)$ is the pheromone value on e_{ij} at the t-th generation, $\eta_{ij} = 1/d_{ij}$ is the heuristic desirability of visiting e_{ij}, β and p_0 are as in Table 4.4, J is a random variable whose probability distribution obeys (4.21) and $p_{ij}(t)$ is the probability that the j-th feasible element will be picked as next node from i at time t.
- Advance to next index $k \leftarrow \text{mod}(k, |\vec{\varphi}_{child}|) + 1$.
- Set $\vec{\varphi}_{child}(k) \leftarrow next$; update robot's cargo $l \leftarrow l - q_{next}$.
- Repeat all but the first step until all elements in $\vec{\varphi}_{child}$ are allocated.

$$next = \begin{cases} \text{argmax}_{j \in F} \left\{ \tau_{ij}(t) \cdot [\eta_{ij}]^{\beta} \right\}, & \text{if } rand( ) \leq p_0 \\ J, & \text{otherwise} \end{cases} \tag{4.20}$$

$$p_{ij}(t) = \begin{cases} \frac{\tau_{ij}(t) \cdot [\eta_{ij}]^{\beta}}{\sum_{x \in F} \left(\tau_{ix}(t) \cdot [\eta_{ix}]^{\beta} \right)}, & \text{if } j \in F \\ 0, & \text{otherwise} \end{cases} \tag{4.21}$$

Notice how, in absence of feasible neighbors in the parent chromosomes for the current offspring position, it is the collective knowledge embedded on the pheromone matrix together with the problem-specific hints of the heuristic function which decide what node should be appended next to the child tour. Because the selected element may not belong to any parent, this crossover operator is said to be of "mixed" type, a successful trend in recent GA research [62].

4.6.6 Generational Pre-Crossover Parent Mutation

A dynamic population renewal mechanism is adopted in GA-ACO. The purpose is to diversify the set of candidate solutions through generational parent mutation so as to bring up possibly neglected pickup nodes into the tour composition.

A parent chromosome $\vec{\varphi}_i$ undergoes mutation before crossover with the following probability:

$$p_M(\vec{\varphi}_i) = \frac{\max(0, \text{age}(\vec{\varphi}_i) - \text{ageIni})}{\text{maturity} - \text{ageIni}} \tag{4.22}$$

where $\text{age}(\vec{\varphi}_i)$ is the age of individual $\vec{\varphi}_i$, defined as the number of generations it has been kept in the population. AgeIni and maturity are user-specified parameters as defined in Table 4.4. The former allows the chromosome to contribute to offspring formation before it is mutated, whereas the latter establishes an aging threshold for the individual to be disposed.

Because probabilistic mutation of an individual only occurs if it has been selected first as crossover parent, then some chromosomes in the population could indeed live longer than "maturity" generations. The *ApplyMutation( )* routine consists of randomly replacing a percentage p of the $|V_D| - Q_0$ pickups in the chromosome with unvisited peers.

Performing a generational mutation at the parent side (as opposed to traditional offspring mutation) ensures a fresh renewal of the population and lowers the stagnation likelihood of the algorithm, as most of the chromosomes will be modified somehow after a certain number of generations. This is a sought-after feature when facing SC^2R instances, where the need to

consider neglected pickup nodes calls for a higher degree of variability in the tour composition.

4.6.7 Handling Pheromone Trails

Once the next population has been formed, the pheromone matrix must be updated. We employ dynamically evolving pheromone bounds τ_{min}, τ_{max} as in [131] to cope with the stagnation phenomenon present in early ACO versions.

At the outset of the hybrid method, all pheromone entries are initialized to τ_{max} and subsequently clamped to the $[\tau_{min}; \tau_{max}]$ interval. They are calculated after (4.23) and (4.24), with parameter ρ being the pheromone persistence rate, as shown in Table 4.4.

$$\tau_{max} = \frac{1}{1-\rho} \cdot \frac{1}{f(S^{gb})} \tag{4.23}$$

$$\tau_{min} = \tau_{max}/(2 \cdot |S^{gb}|) \tag{4.24}$$

where $f(\cdot)$ is the fitness function (tour cost) and S^{gb} is the globally best solution so far.

At the end of every generation, *UpdatePheromoneTrails( )* modifies all entries in the pheromone matrix as shown in (4.25).

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij} \tag{4.25}$$

where

$$\Delta\tau_{ij} = \begin{cases} 1/f(S^{ib}), & \text{if } e_{ij} \in S^{ib} \\ 1/f(S^{gb}), & \text{if } e_{ij} \in S^{gb} \\ 0, & \text{otherwise} \end{cases}$$

with S^{ib} being the best tour in the current generation.

This pheromone update scheme yielded better results than that of MMAS in Sect. 4.3.3.

Table 4.5: Sampling Range for Robot Information

Robot Property	Range
Robot Capacity (X_{rc})	1 - 5
Initial Robot Load (X_{irl})	0 - Robot Capacity
CPU Time Available (X_{cpu})	5 - 600 seconds

4.6.8 Termination Criterion

GA-ACO terminates when it reaches either a maximum number of generations or an upper bound of the running time. The latter seems more realistic due to the responsiveness constraints placed upon the WSRN in real time.

4.7 GA-ACO Experimental Study

This section describes the layout of the simulation setup, the evaluation methodology and the obtained empirical results.

4.7.1 Simulation Setup

All experiments in this study are conducted in MATLAB R2009b on an Intel Core i7 CPU 860 @ 2.80 GHz with 6 GB of RAM under Windows 7 Home Premium with a 64-bit architecture. They consist of synthetic scenarios, each holding information about the problem description (robot and sensor network properties) and the algorithm's parameters. Robot properties are the robot capacity X_{rc}, the initial robot load X_{irl} and CPU time X_{cpu} available to compute a solution. Table 4.5 shows the sampling range for each property.

Sensor network properties are the total number of sensors X_{ns} and the percentage of delivery nodes X_{dn}. Table 4.6 presents the sampling range for these properties. The node set consists of at most 25% delivery nodes, leading to unbalanced pickup and delivery problem instances. Therefore, the selection of the appropriate set of pickup locations is highly important. Once the values of the sensor network properties are known, the sensors are spatially

Table 4.6: Sampling Range for Network Information

Network Property	Range
Total number of sensors (X_{ts})	30 - 500
Percentage of delivery nodes (X_{dn})	5% - 25%

Table 4.7: Spatial Distributions of the Sensor Network

Distribution	Depot δ	Delivery Node d_i	Pickup Node p_i
1	$(0,0)$	U	U
2	$(0,0)$	U	$\mathcal{N}(d_i, \sigma^2)$
3	$(0,0)$	$\mathcal{N}(\delta, \sigma^2)$	$\mathcal{N}(\delta, \sigma^2)$
4	$(0,0)$	$\mathcal{N}(\delta, \sigma^2)$	$\mathcal{N}(d_i, \sigma^2)$
5	U	U	U
6	U	U	$\mathcal{N}(d_i, \sigma^2)$
7	U	$\mathcal{N}(\delta, \sigma^2)$	$\mathcal{N}(\delta, \sigma^2)$
8	U	$\mathcal{N}(\delta, \sigma^2)$	$\mathcal{N}(d_i, \sigma^2)$
9	$\mathcal{N}(r, \sigma^2)$	$\mathcal{N}(r, \sigma^2)$	$\mathcal{N}(r, \sigma^2)$

distributed in a virtual field of $1,000 \times 1,000$ square units. First, the depot's location δ is chosen either deterministically at the field center $(0,0)$, or randomly following a bivariate uniform distribution U, or stochastically following a bivariate normal distribution $\mathcal{N}(r, \sigma^2)$ with r being a uniform random 2D point. Second, the coordinates d_i of the delivery nodes are set either randomly following U, or stochastically after $\mathcal{N}(\delta, \sigma^2)$ centered around the depot, or stochastically following $\mathcal{N}(r, \sigma^2)$ and centered around the random point r. Finally, the pickup nodes are positioned either randomly after U, or stochastically following $\mathcal{N}(d_i, \sigma^2)$ and surrounding the coordinates d_i of an arbitrary sensing hole, or stochastically through $\mathcal{N}(\delta, \sigma^2)$ centered around the depot, or stochastically via $\mathcal{N}(r, \sigma^2)$ and centered around r. These degrees of freedom give rise to nine different spatial distributions, which are summarized in Table 4.7. Nodes generated outside the field border are redrawn until they lie within, thus giving rise to censored distributions.

A 1-TSP-SELPD problem instance contains information about the robot, the sensor network and its spatial distribution. To complete a data scenario, knowledge of the GA-ACO

Table 4.8: Sampling Range for Heuristic Information

Heuristic Information	Range
pop_size (X_{pop})	1 - 60
cl (X_{cls})	1 - 30
p_{NFC} (X_{nfc})	0 - 100
p_{CR} (X_{cp})	1 - 99
β (X_{β})	1 - 10
p_0 (X_{p0})	0 - 100
ageIni parent mutation (X_{iapm})	1 - 20
maturity parent mutation (X_{mpm})	ageIni - 20
Percentage p parent mutation (X_{ppm})	0 - 100
ρ (X_{ρ})	1 - 99
Or-OPT to improve initial population (X_{ori})	{0,1}
PEX to improve initial population (X_{pxi})	{0,1}
2-OPT to improve initial population (X_{2i})	{0,1}
Parent mutation (X_{pm})	{0,1}
Pheromone knowledge during crossover (X_{pkx})	{0,1}
Or-OPT to improve crossover (X_{orx})	{0,1}
PEX to improve crossover (X_{pxx})	{0,1}
Two-OPT to improve crossover (X_{2x})	{0,1}

instance is required as well. This comes in the form of ten built-in parameters (see Table 4.4) and eight heuristic elements which can be turned on/off. They are all depicted in Table 4.8 along with the corresponding sampling ranges.

4.7.2 Evaluation Methodology

The methodological approach governing the empirical assessment consists of three stages. First, a regression analysis is used to determine GA-ACO's optimal parameter values and to construct decision rules stating which heuristic elements should be activated for a particular problem instance. Three algorithmic strategies are defined: the *full strategy*, which activates all heuristic elements by default; the *generic strategy*, which incorporates a fixed, optimized set of heuristic elements; and the *intelligent strategy*, which dynamically turns on an optimized group of heuristic components given the problem instance. Second, the performances of these

three strategies are statistically contrasted to determine the winner(s). Finally, results yielded by the winner(s) are statistically compared against those of a customized Variable Neighborhood Search (VNS) and the MMAS heuristic in Section 4.3, which is the only scheme in literature we know of that targets 1-TSP-SELPD so far.

For the regression analysis, 400 problem instances are created by drawing random values from the sampling ranges for robot and network information (Tables 4.5 and 4.6) and by randomly picking a spatial distribution from Table 4.7. Then, for each problem instance, 20 scenarios are generated by adding the required algorithmic information, which is randomly sampled from the corresponding intervals (Table 4.8). Results of further comparative studies with this research must be handled with care when extrapolated to problem instances and algorithmic configurations outside these sampling ranges. Overall, 8,000 scenarios are randomly defined[2], each describing both the problem and the algorithmic structure. Every scenario is run once and the cost of the best solution found is recorded.

A regression analysis on these 8,000 records is carried out, in which the dependent variable y_i represents the cost of the best tour found for a single run of a specific scenario i and $\vec{X}_i$ denotes the vector of the values of the independent (algorithmic) variables for the i-th scenario. This allows to assess the impact of parameter values and heuristic components on the solution quality. However, data come from a population with a hierarchical structure and the sampling proceeds in two phases: first, a sample of 400 problem instances is drawn, and next a sample of 20 scenarios is defined per problem instance. In such samples, the assumption of independence of the observations is most likely violated, which makes the estimates of the standard errors of conventional statistical tests such as classical linear regression too small [74]. Therefore, a *multilevel regression model*, also known as *random or mixed effect model* is used. One of the main structural differences between classical regression and multilevel regression is the presence of multiple variance components in the latter. Our analysis will contain a component $\eta_{j[i]}$ which captures the unexplained variance related to the problem instance j of scenario i

[2]All simulation data are available at `http://www.eecs.uottawa.ca/~rfalc032/files/IEEE-TEC-GA-ACO-data.zip`

and a component ϵ_i which captures the unexplained variance related to scenario i in particular. This leads to the formulation in (4.26), with $\eta_j \sim \mathcal{N}(0, \sigma_\alpha^2)$ and $\epsilon_i \sim \mathcal{N}(0, \sigma_y^2)$. The reader interested in the more details on the mixed effect model is directed to [61].

$$y_i = X_i\beta + \eta_{j[i]} + \epsilon_i \tag{4.26}$$

The second and third stages of the empirical study contrast three different algorithmic strategies and attempt to uncover significant differences among them in terms of solution quality. The approach suggested by [37] is applied in this article. For each experiment, 40 new problems are generated within the robot and network information sampling ranges. Each strategy solves all 40 problems ten times and the Friedman test is applied to detect statistically significant differences among the average performance of the strategies. The Friedman test is a non-parametric equivalent of the repeated-measures ANOVA test, but is based on the ranking of the strategy per data set instead of the quantitative cost of the best solution found. More specifically, the different strategies are applied to each problem instance ten times and the average cost over the ten runs is stored for each strategy. They are next ranked from best to worst for each problem instance and the Friedman test analyzes the statistical significance of the difference in average rank over all problem instances among the different strategies. If according to the Friedman test, statistical differences between the strategies exist, the Nemenyi test is used to compare the strategies pairwise. The Nemenyi test provides a control mechanism for family-wise error in multiple hypothesis testing and is similar to the Tukey test for ANOVA. For more details on Friedman and Nemenyi tests, see Demšar [37]. The author discusses why the ANOVA test is inappropriate for comparing multiple classifiers, but the same criticism equally holds for optimization methods.

4.7.3 Experiment 1: Parameter Setting and Decision Rules

A multilevel regression analysis is performed on a set of 8,000 data scenarios to discover optimal values for the algorithm's parameters and to generate decision rules. The cost of the

best solution found in the i-th scenario is set as the dependent variable y_i while the independent variables X_i refer to the robot properties (Table 4.5), the network properties (Table 4.6), the algorithm's parameters and the heuristic components (Table 4.8). Each heuristic element is modeled as a dummy (binary) variable (e.g. X_{pm} for parent mutation). Should that heuristic component have any associated built-in parameter (e.g. the X_{iapm} parameter only relates to parent mutation heuristic element X_{pm}), their interaction is captured in the model ($X_{pm}X_{iapm}$). The same is true for the interactions between the heuristic component and the variables representing robot or network properties, e.g. ($X_{pm}X_{rc}$). Furthermore, squared values of variables representing robot or network properties and algorithmic parameters are also added to the regression model to consider non-linear effects.

Finally, to facilitate the interpretation of the regression coefficients, various transformations on the predictor variables are enforced. The percentage of delivery nodes (X_{dn}) is transformed into the absolute number of delivery nodes. The CPU time available (X_{cpu}) is divided by 60 to transform the scale from seconds to minutes. Robot and network variables are centered around their sample mean shown in Table 4.9 and the heuristic parameter variables are centered around the midpoint of their corresponding sampling range in Table 4.8. Finally, the maturity parameter (X_{mpm}) is transformed into the difference between the ageIni parameter and the original maturity parameter. This transformation reduces the collinearity which exists between maturity and ageIni. Unless explicitly stated, all notation hereafter refers to the transformed variables. The regression coefficients are displayed in Table 4.10. Insignificant squared variables were removed from the model.

Table 4.9: Sampling Means for Network and Robot Information

Variable	Sampling Mean
Total number of sensors (X_{ts})	263.30
Number of delivery nodes (X_{dn})	38.25
Robot Capacity (X_{rc})	3.15
Initial Robot Load (X_{irl})	1.55
CPU Time Available (X_{cpu})	5.01

Table 4.10: Multilevel regression analysis. The intercept is denoted by (*)

Variable	Coef.	St.Err.	p-value	Variable	Coef.	St.Err.	p-value	Variable	Coef.	St.Err.	p-value
(*)	5076.4	94.92	0.00	X_{2i}	-37.29	36.10	0.30	X_{pxx}	-343.83	30.66	0.00
X_{ts}	-1.51	1.02	0.14	$X_{2i}X_{ts}$	-0.17	0.20	0.41	$X_{pxx}X_{ts}$	-0.08	0.20	0.67
X_{dn}	101.18	5.31	0.00	$X_{2i}X_{dn}$	13.14	1.20	0.00	$X_{pxx}X_{dn}$	-7.56	1.20	0.00
X_{rc}	-234.59	76.56	0.00	$X_{2i}X_{dn}^2$	0.21	0.02	0.00	$X_{pxx}X_{dn}^2$	0.08	0.02	0.00
X_{irl}	-62.52	77.61	0.42	$X_{2i}X_{rc}$	-90.86	14.42	0.00	$X_{pxx}X_{rc}$	81.74	14.39	0.00
X_{cpu}	-27.62	31.9	0.38	$X_{2i}X_{rc}^2$	55.43	10.35	0.00	$X_{pxx}X_{rc}^2$	-30.96	10.37	0.00
X_{pop}	-3.13	0.5	0.00	$X_{2i}X_{irl}$	6.64	14.59	0.65	$X_{pxx}X_{irl}$	-3.96	14.59	0.79
X_{pop}^2	0.40	0.03	0.00	$X_{2i}X_{cpu}$	-23.15	5.98	0.00	$X_{pxx}X_{cpu}$	-26.71	5.96	0.00
X_{cls}	-7.27	2.31	0.00	$X_{2i}X_{cls}$	0.67	2.09	0.75	$X_{pxx}X_{cls}$	0.04	2.07	0.98
X_{cls}^2	0.83	0.19	0.00	$X_{2i}X_{cls}^2$	1.04	0.27	0.00				
X_{cp}	-0.82	0.3	0.01					X_{2x}	175.37	22.9	0.00
X_{cp}^2	0.05	0.01	0.00	X_{pm}	29.07	34.59	0.40	$X_{2x}X_{ts}$	-0.22	0.20	0.28
X_{nfc}	-3.39	0.29	0.00	$X_{pm}X_{ts}$	0.16	0.19	0.40	$X_{2x}X_{dn}$	5.71	1.19	0.00
				$X_{pm}X_{dn}$	-0.34	1.00	0.73	$X_{2x}X_{dn}^2$	-0.08	0.02	0.00
X_{ori}	63.16	22.83	0.01	$X_{pm}X_{rc}$	12.74	14.44	0.38	$X_{2x}X_{rc}$	0.99	14.39	0.94
$X_{ori}X_{ts}$	-0.22	0.20	0.27	$X_{pm}X_{rc}^2$	-30.27	10.38	0.00	$X_{2x}X_{irl}$	-14.38	14.57	0.32
$X_{ori}X_{dn}$	5.07	1.20	0.00	$X_{pm}X_{irl}$	15.35	14.53	0.29	$X_{2x}X_{cpu}$	9.81	5.98	0.10
$X_{ori}X_{dn}^2$	0.05	0.02	0.03	$X_{pm}X_{cpu}$	-6.39	5.96	0.29	$X_{2x}X_{cls}$	4.12	2.07	0.05
$X_{ori}X_{rc}$	-38.11	14.41	0.01	$X_{pm}X_{iapm}$	-5.74	2.74	0.04				
$X_{ori}X_{irl}$	3.58	14.56	0.80	$X_{pm}X_{iapm}^2$	0.99	0.41	0.02	X_{pkx}	21.12	22.36	0.35
$X_{ori}X_{cpu}$	-24.47	5.98	0.00	$X_{pm}X_{mpm}$	-2.68	3.54	0.45	$X_{pkx}X_{ts}$	-0.02	0.19	0.90
				$X_{pm}X_{ppm}$	-0.08	0.41	0.83	$X_{pkx}X_{dn}$	3.24	1.00	0.00
X_{pxi}	-170.58	26.98	0.00					$X_{pkx}X_{rc}$	18.83	14.35	0.19
$X_{pxi}X_{ts}$	0.10	0.19	0.62	X_{orx}	-85.31	22.88	0.00	$X_{pkx}X_{irl}$	9.60	14.60	0.51
$X_{pxi}X_{dn}$	-4.20	1.01	0.00	$X_{orx}X_{ts}$	0.22	0.20	0.28	$X_{pkx}X_{cpu}$	2.19	6.01	0.72
$X_{pxi}X_{rc}$	26.81	14.37	0.06	$X_{orx}X_{dn}$	0.82	1.20	0.49	$X_{pkx}X_{\beta}$	-20.20	4.26	0.00
$X_{pxi}X_{irl}$	-5.98	14.50	0.68	$X_{orx}X_{dn}^2$	0.10	0.02	0.00	$X_{pkx}X_{\beta}^2$	6.29	1.70	0.00
$X_{pxi}X_{cpu}$	-1.86	6.01	0.76	$X_{orx}X_{rc}$	-61.86	14.38	0.00	$X_{pkx}X_{\rho}$	0.14	0.43	0.74
$X_{pxi}X_{cpu}^2$	5.36	2.44	0.03	$X_{orx}X_{irl}$	2.90	14.51	0.84	$X_{pkx}X_{\rho 0}$	-2.03	0.42	0.00
$X_{pxi}X_{cls}$	0.92	2.07	0.65	$X_{orx}X_{cpu}$	-17.55	5.97	0.00				

Table 4.10 reveals that for the 8,000 studied scenarios an average cost of 5,076 is found. Of all the problem-specific variables, only the number of delivery nodes and the robot capacity have a statistically significant main effect of respectively 101.18 and -234.59. These results confirm the expectation that additional delivery nodes increase the tour length while additional robot capacity decreases the tour length.

The regression coefficients also reveal that various parameters have a non-linear effect on the tour length, indicating the existence of an optimal parameter value which minimizes the tour length. This optimal value can be determined analytically using first and second-order derivatives which results in the following optimal values for the untransformed parameters: a population size of 30, a crossover probability of 58, an initial mutation age of 13, a β in the pheromone step of 7 and a candidate list of size 16. Note that for the latter, only the significant interaction effects were considered; except when the second-order term is significant, in which

case the first-order term is always considered irrespective of its significance level.

The parameters p_{NFC} and p_0 appear to have a linear effect, which results in the optimal values of $p_{NFC} = 100$ and $p_0 = 100$ for the untransformed parameters. A p_{NFC} value of 100 implies that the second step of population initialization should be fully deterministic and the nearest feasible customer should always be selected as the next customer. The p_0 value of 100 also reduces the level of randomness in our heuristic as it implies that the next element in the tour during the informed, incremental crossover should be based on greedy selection and not selected randomly. These two results suggest that the algorithm already contains sufficient stochasticity through step 1 of the population initialization, the parent mutation and the local search operators. Apparently, adding additional stochastic behavior through p_{NFC} and p_0 only deteriorates the effectiveness of the heuristic.

Finally, the parameters maturity X_{mpm} and mutation percentage X_{ppm} from the parent mutation step and ρ from the pheromone step showed no statistical significant effect. Therefore, some educated guesses had to be made to determine their values. For the untransformed ρ parameter, a value of 95 is suggested which is in accordance with the existing literature. As for the two non-significant untransformed parent mutation parameters, we suggest to set them at a median value, which corresponds to $X_{mpm} = 16$ and $X_{ppm} = 50$.

Regressions coefficients also allow to predict the impact of a particular heuristic element on the solution quality of a given problem instance. For example, to determine the effect of the Or-OPT local search operator during population initialization, one has to focus on the X_{ori} regression term and all its associated regression terms. To make a conservative estimate of the heuristic element's bearing on the dependent variable, only the terms significant at 5% are considered because the uncertainty about the true sign of the insignificant terms is too high. This results in the decision rules portrayed in Table 4.11.

These decision rules give rise to the definition of three GA-ACO algorithmic strategies, i.e. the *full* (FULL) strategy, the *generic* (GEN) strategy and the *intelligent* (INT) strategy. They all share the same set of optimal parameter values, yet differ in which heuristic ele-

Table 4.11: Decision rules governing the activation of heuristic elements in GA-ACO

Antecedent	Condition	Consequent
$63.16 + 5.07X_{dn} + 0.05X_{dn}^2 - 38.11X_{rc} - 24.47X_{cpu}$	< 0	$\Rightarrow X_{ori} = 1$
$-170.58 - 4.2X_{dn} - 1.86X_{cpu} + 5.36X_{cpu}^2$	< 0	$\Rightarrow X_{pxi} = 1$
$13.14 * X_{dn} + 0.21X_{dn}^2 - 90.86X_{rc} + 55.43X_{rc}^2 - 23.15X_{cpu} + 0.67X_{cls} + 1.04X_{cls}^2$	< 0	$\Rightarrow X_{2i} = 1$
$12.74X_{rc}30.27X_{rc}^2 - 5.74X_{iapm} + 0.99X_{iapm}^2$	< 0	$\Rightarrow X_{pm} = 1$
$3.24X_{dn} - 20.2X_{\beta} + 6.29X_{\beta}^2 - 2.03X_{p0}$	< 0	$\Rightarrow X_{pkx} = 1$
$-85.31 + 0.82X_{dn} + 0.1X_{dn}^2 - 61.86X_{rc} - 17.55X_{cpu}$	< 0	$\Rightarrow X_{orx} = 1$
$-343.88 - 7.56X_{dn} + 0.08X_{dn}^2 + 81.74X_{rc} - 30.96X_{rc}^2 - 26.71X_{cpu}$	< 0	$\Rightarrow X_{pxx} = 1$
$175.37 + 5.71X_{dn} - 0.08X_{dn}^2 + 4.12X_{cls}$	< 0	$\Rightarrow X_{2x} = 1$

ments are activated. The FULL strategy turns on all heuristic elements. The GEN strategy only activates the PEX operator during population initialization, the parent mutation step, the use of pheromone knowledge during crossover and the Or-OPT and PEX operators during crossover. These are the five elements for which the decision rules in Table 4.11 fire in case of an "average" problem, i.e. a problem instance having sample mean values for the robot and network properties. Note that after transformation, these variables equal zero. The INT strategy uses the robot and network information in the problem instance, transforms them by centering around the values in Table 4.9 and checks the set of decision rules in Table 4.11. Only the heuristic elements whose rule fire are activated.

4.7.4 Experiment 2: Comparison of Algorithmic Strategies

In order to contrast the three GA-ACO strategies, 40 new problem instances are generated and the Friedman test is conducted as described in section 4.7.2. Friedman's χ^2 statistic ($\chi_F^2 = 41.15$ with 2 degrees of freedom) reveals that there are statistically significant differences in performance between the three strategies at a 1% significance level. Note that performance is defined as the expected objective function value of running the strategy on a particular problem instance. Additionally, Fig. 4.5 shows the average ranking of the three strategies, which clarifies that INT ranks on average as the best strategy, FULL ranks on average as the worst strategy and GEN lies in between. Strategies which are not statistically significant at a 1% significance level are connected with a horizontal line on Fig. 4.5, which shows that while

the INT approach performs better than the GEN approach, the statistical evidence is lacking to generalize this finding to any arbitrary SC^2R problem instance. On the other hand, Fig. 4.5 also reveals that both the GEN and INT approaches statistically outperform the FULL approach, which proves that the parameter optimization from the previous section was successful.

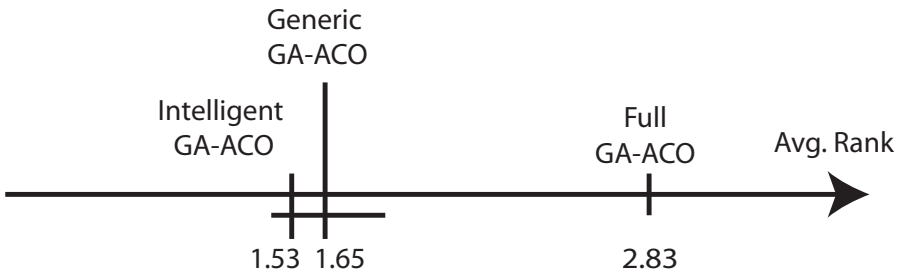


Figure 4.5: Average ranking of the three GA-ACO approaches.

From these results, we conclude that for problem instances which fall within the boundaries under study in this paper, the FULL strategy should never be applied. When problem-specific knowledge is not available to optimize GA-ACO, GEN provides a good alternative to INT, which is indeed preferable when such information is known a priori.

4.7.5 Experiment 3: Comparison versus MMAS and VNS

The last stage of this empirical study compares the performance of GA-ACO's GEN and INT strategies against a Variable Neighborhood Search (VNS) (which consists of the three local search operators used by GA-ACO) and the Max-Min Ant System (MMAS) approach in [52] (Section 4.3). The VNS approach used all three operators, i.e. OR-OPT, PEX and 2-OPT, in a loop and switches the current operator each time it does not find any improvement after five iterations. The VNS implementation serves as a benchmark to test whether the GA-ACO approach performs better than the local search operators it incorporates. Comparison with MMAS verifies whether GA-ACO improves the state-of-the-art in solving SC^2R problems.

Again, 40 problem instances are generated and the Friedman test is conducted as described in Section 4.7.2. Friedman's χ^2 statistic (χ^2_F = 97.73 with 3 degrees of freedom) reports statis-

tically significant performance differences among the four algorithms at 1% significance level. Based on the average ranking depicted in Fig. 4.6 and the outcome of the Nemenyi test, it is easy to realize that VNS is outperformed by both MMAS and the GA-ACO implementations. Furthermore, MMAS is defeated by both the generic GA-ACO and the intelligent GA-ACO. As in the previous section, the latter slightly excels the former, but not in a statistically significant way, which is indicated in Fig. 4.6 by the horizontal bar connecting both approaches.

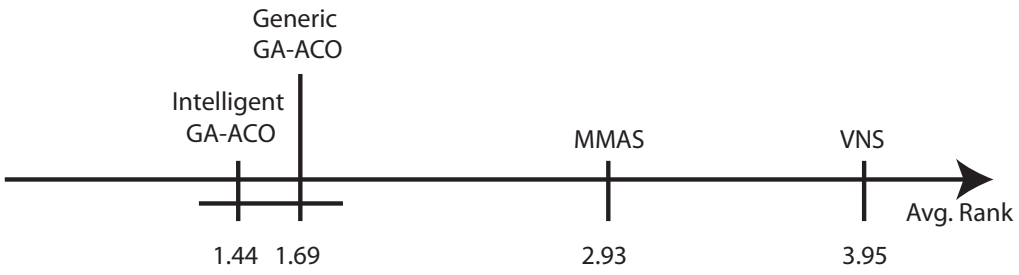


Figure 4.6: Average ranking GA-ACO's GEN and INT variants versus MMAS and VNS.

Table 4.12 gives an insight into why GA-ACO performs better than MMAS. The 40 problem instances used for the analysis are listed. The first six columns describe the different robot and network properties. The next four columns mention the expected cost for GEN, INT, MMAS and VNS, estimated by the average cost over 10 runs. Results demonstrate that the expected costs vary greatly, with 750 as minimum and 12,822 as maximum.

The ranking of the four algorithms per problem instance is displayed next. VNS appears almost always as the worst scheme and MMAS usually comes in third, while INT has a slightly better average ranking than GEN. To get an indication of how good each algorithm performs, their average expected cost, i.e. tour length, is depicted at the bottom of Table 4.12. On average, the INT and GEN approach produce solutions with tour lengths that are respectively 985 and 895 units shorter than those produced by VNS. Clearly, both GA-ACO approaches are more successful in finding good solutions than the collection of local search heuristics they implement. The last three columns show the relative improvement, i.e. the absolute improvement divided by the VNS expected cost. On average, GEN and INT improve VNS by 16% and 17% respectively.

Table 4.12: GA-ACO versus MMAS and VNS algorithms

Problem Instance Descriptors						Expected Cost				Rank				% Improv. wrt VNS		
ID	X_{ts}	X_{dn}	X_{rc}	X_{irl}	X_{cpu}	GEN	INT	MMAS	VNS	GEN	INT	MMAS	VNS	MMAS	GEN	INT
1	119	12	4	4	391	1605	1599	1662	1680	2	1	3	4	-0.01	-0.04	-0.05
2	125	11	5	4	416	2923	2921	2988	3115	2	1	3	4	-0.04	-0.06	-0.06
3	125	24	1	0	34	5366	5278	5395	6928	2	1	3	4	-0.22	-0.23	-0.24
4	127	20	3	1	110	1276	1294	1399	1597	1	2	3	4	-0.12	-0.20	-0.19
5	129	32	5	5	474	5544	5553	5978	6690	1	2	3	4	-0.11	-0.17	-0.17
6	131	25	3	3	179	2962	2986	3303	3596	1	2	3	4	-0.08	-0.18	-0.17
7	139	19	4	0	359	2891	2886	3048	3300	2	1	3	4	-0.08	-0.12	-0.13
8	139	33	2	2	47	2422	2403	2558	2843	2	1	3	4	-0.10	-0.15	-0.15
9	143	36	5	4	7	3214	3298	3379	3533	1	2	3	4	-0.04	-0.09	-0.07
10	145	26	5	3	9	5464	5730	5968	5872	1	2	4	3	0.02	-0.07	-0.02
11	168	12	4	2	139	1098	1118	1167	1227	1	2	3	4	-0.05	-0.11	-0.09
12	184	20	1	0	278	1926	1917	2066	2601	2	1	3	4	-0.21	-0.26	-0.26
13	200	20	4	1	420	1314	1316	1435	1622	1	2	3	4	-0.12	-0.19	-0.19
14	203	26	3	3	441	4240	4301	4590	5720	1	2	3	4	-0.20	-0.26	-0.25
15	209	15	3	0	323	1892	1916	2107	2355	1	2	3	4	-0.11	-0.20	-0.19
16	245	12	4	4	115	750	774	777	863	1	2	3	4	-0.10	-0.13	-0.10
17	246	12	4	3	45	1567	1628	1655	1764	1	2	3	4	-0.06	-0.11	-0.08
18	55	3	5	1	312	1324	1324	1334	1381	1.5	1.5	3	4	-0.03	-0.04	-0.04
19	80	16	3	3	477	2290	2284	2328	2578	2	1	3	4	-0.10	-0.11	-0.11
20	91	19	1	1	252	1996	1951	1974	2701	3	1	2	4	-0.27	-0.26	-0.28
21	267	13	4	2	586	2920	2931	3093	3344	1	2	3	4	-0.08	-0.13	-0.12
22	279	50	5	3	77	3070	3069	3178	3384	2	1	3	4	-0.06	-0.09	-0.09
23	281	67	1	1	154	9486	8874	9238	11925	3	1	2	4	-0.23	-0.20	-0.26
24	292	15	5	5	79	3468	3636	3626	4385	1	3	2	4	-0.17	-0.21	-0.17
25	302	76	5	4	176	5919	5889	6073	6693	2	1	3	4	-0.09	-0.12	-0.12
26	306	52	5	3	535	5106	5136	5587	6088	1	2	3	4	-0.08	-0.16	-0.16
27	312	22	4	1	39	2293	2333	2449	2433	1	2	4	3	0.01	-0.06	-0.04
28	318	25	1	0	405	3608	3563	4010	5274	2	1	3	4	-0.24	-0.32	-0.32
29	336	20	4	0	196	2148	2126	2308	2477	2	1	3	4	-0.07	-0.13	-0.14
30	339	78	1	1	227	7106	6558	6842	8934	3	1	2	4	-0.23	-0.20	-0.27
31	344	83	2	2	491	9946	9323	10016	12822	2	1	3	4	-0.22	-0.22	-0.27
32	371	78	5	5	590	6227	6176	6470	7335	2	1	3	4	-0.12	-0.15	-0.16
33	374	49	5	5	423	6485	6490	7080	7765	1	2	3	4	-0.09	-0.16	-0.16
34	376	38	1	1	472	5464	5423	6282	8839	2	1	3	4	-0.29	-0.38	-0.39
35	388	85	2	1	326	9805	8935	9608	12240	3	1	2	4	-0.22	-0.20	-0.27
36	391	94	5	0	128	10409	10043	10872	11587	2	1	3	4	-0.06	-0.10	-0.13
37	419	67	3	1	198	8272	7721	8391	9397	2	1	3	4	-0.11	-0.12	-0.18
38	444	27	4	3	318	4843	4839	5207	5659	2	1	3	4	-0.08	-0.14	-0.14
39	455	96	4	0	592	6689	6224	6880	7784	2	1	3	4	-0.12	-0.14	-0.20
40	458	50	1	0	445	2680	2644	2714	3478	2	1	3	4	-0.22	-0.23	-0.24
					Average	4200	4110	4376	5095	1.69	1.44	2.93	3.95	-11.96	-16.14	-16.70

Compared against MMAS, the results also show that both GA-ACO approaches perform statistically significantly better than MMAS. Both the GEN and INT approaches provide a relative improvement of 5% in comparison with MMAS. A more detailed investigation of the results in Table 4.12 reveal that neither the GEN nor the INT approach ever performs worse than MMAS and the maximum relative improvement within the group of 40 optimization problems studied goes up to 14% and 13% for the INT and GEN approach respectively. These results clearly show that the GA-ACO method is preferable over the MMAS approach.

4.7.6 Algorithmic Complexity

To assess the time complexity of the GEN and FULL variants of GA-ACO, an upper bound will be derived. The INT version is not included in the analysis because of its highly problem-specific nature. Table 4.13 summarizes the temporal complexity of GA-ACO's building blocks in terms of the total number of nodes (N) and the cardinality of the population (pop_size).

Table 4.13: Worst-case time complexity of GA-ACO's algorithmic components

Line	Algorithmic Component	Upper bound
1	*ComputeCandidateList( )*	$N^2 \log_2 N$
2	*ComputeCandidatePickupList( )*	$N^2/4$
4	*GenerateIndividual( )*	$N^2/4 + 3N/4$
5	*ImproveIndividual( )::Or-OPT*	$N^2/4 - N/2$
5	*ImproveIndividual( )::PEX*	$N^2/16 + N/2$
5	*ImproveIndividual( )::2-OPT*	$N^2/4 - 3N/2 + 2$
7	*EvaluatePopulation( )*	$N/2 \times$ pop_size
11	*SelectCrossoverParents( )*	4pop_size + 2
13	*ApplyMutation( )*	$5N/4$
15	*ApplyCrossover( )*	N^2
20	*UpdatePopulation( )*	pop_size $\log_2$ pop_size + pop_size/2
22	*UpdatePheromoneTrails( )*	N^2

After setting pop_size equal to 60 (the maximum value explored in this study, see Table 4.8) and expressing the main loop of lines 8–23 as a function of the maximum number of generations M, the worst-case time complexity of FULL GA-ACO is $O(25MN^2 + N^2(50 +$

$\log_2 N$)).

After an identical rationale, the upper bound for GEN GA-ACO is derived as $O(21MN^2 + N^2(20 + \log_2 N))$. Because the GEN strategy turns off three expensive, not-so-useful steps with which the FULL strategy still has to deal, it is able devote more time to crossover and mutation, hence yielding remarkably superior results.

Both GA-ACO variants are more complex than VNS, which bears an $O(MN^2 + N^2(1 + \log_2 N))$. MMAS is the most time-consuming method with $O(15MN^3 + 76.5MN^2)$ given the tour construction procedure for each ant in the colony, which takes place iteration-wise.

4.8 Conclusions

This chapter sheds light on a brand new problem arising in WSRNs, *carrier-based coverage repair*, casts its single-robot case (SC^2R) into the realm of combinatorial optimization and puts forward the first two known approaches that tailor their nature-inspired metaheuristic design to SC^2R landscapes in order to yield high-quality solutions in a short running time.

The first scheme is based on a successful ant colony model (MMAS) which overcomes the stagnation phenomenon endured by early ACO approaches. It renders good solutions even though little adjustments were made to its optimization machinery in order to fit the problem at hand. The two main lessons we learned are: (1) that the heuristic function plays a much more beneficial role if several perspectives on the local decision neighborhood are integrated into it and (2) that the two-step, pheromone-driven exploration strategy in MMAS alleviated to a large extent the drawbacks caused by increased dimensionality of the search space.

Later on, the amalgamation of evolutionary and swarm intelligence representatives (GA and ACO, respectively) proved to excel the MMAS protocol at 1% significance level throughout a wide assortment of synthetic data scenarios, this due to the solution improvement via refined initialization and local search, the high degree of exploration provided by generational parent mutation and the presence of the pheromone trails in the incremental crossover oper-

ator. Furthermore, three algorithmic strategies for the application of the hybrid method are delineated on the basis of the activation of particular heuristic elements. Our hybrid approach also beat a purely local-search-based implementation (VNS), thus confirming that its combinatorial optimization machinery largely rests upon the successful synergy between genetic algorithms and artificial ant colonies.

Future research efforts in the CBCR arena are briefly described in Chapter 8.

In the next chapter, we extend the SC^2R problem to the case in which a robotic fleet is available at the base station for periodical sensor relocation purposes. Then a hybrid metaheuristic approach that runs in a centralized fashion is brought forward.

CHAPTER 5

MULTIPLE-CARRIER COVERAGE REPAIR

5.1 Introduction

The previous chapter acquainted the reader with a pragmatic and conceptually novel robot-assisted coverage repair scenario in a WSN-monitored environment. It was termed *single-carrier coverage repair* (SC^2R) and two centralized solution approaches relying on the concurrent exploration capacity of biologically-inspired metaheuristic optimization algorithms were presented and their performance discussed at length.

The extension of the SC^2R framework in Chapter 4 to multiple robots seems both natural and desirable. From the perspective of the WSN manager, a small robotic fleet could still be purchased with a fairly modest budget for network infrastructure and it will handle the issue of sensor replacement much faster than if the task is delegated to a single robot. We refer to this new generalization as the *multiple-carrier coverage repair* problem (MC^2R) and devote this chapter to its analytical and empirical study.

MC^2R can be tackled from both centralized and localized standpoints. The former setting relies on the same set of assumptions described in Sect. 4.1 except that now multiple robots lie at the base station, ready to engage in sensor relocation operations. This readiness to shorten the network-wide coverage repair latency by having them all out working in the field is also

balanced by an awareness of any emergency call they may need to respond to after the current coverage repair operation is initiated. This means that not all robots will likely be involved in replacing damaged sensors unless strictly necessary. Hence, priority will be given to route plans involving fewer robots as long as the repair latency remains within tolerable bounds.

In a localized implementation, robots are not together at the base station but rather roaming individually (either across the entire deployment field or confined within their own virtual subregion) and receiving in-situ reports from nearby sensors about damaged sensing units in their area of influence. A collaborative scheme is to be enforced so that the robotic agents may help each other replace the faulty sensing nodes if needed as promptly as possible while consuming the least amount of resources. We envision that the localized scheme could rely to a large extent on any of the centralized algorithms put forth in Chapter 4 to figure out the individual robot replacement trajectory for a particular network subregion. However, we restrict ourselves in this chapter to address the centralized MC^{2}R case and leave the localized version for further research endeavours.

Sect. 5.2 unpacks the mathematical formalization of centralized MC^{2}R as a particular case of a brand new combinatorial optimization problem (*the one-commodity vehicle routing problem with selective pickup and delivery*, 1-VRP-SELPD). The reader is directed to Sect. 2.2.3 for widening her understanding of 1-VRP-SELPD. Then a hybrid metaheuristic that takes advantage of the synergy between firefly swarms and harmony search is thoroughly explained in Sect. 5.3 and its computational performance assessed in Sect. 5.4 by means of extensive simulations. Conclusions are drawn at the end of the chapter.

5.2 MC^{2}R Problem Formulation

In this section, the MC^{2}R problem is formally modeled as a unitary 1-VRP-SELPD instance. Any MC^{2}R scenario can be represented by a tuple $T = \langle G, \vec{q}, R, Q_0, Q \rangle$. A robot team $R = \{r_1, r_2, ...r_m\}$ of exactly m agents, each with initial cargo Q_0 and capacity Q, has to pick up

and deliver sensors in a network represented by a graph G and a demand vector $\vec{q}$.

The sensor network is described by a complete graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex set and E is the edge set, with e_{ij} being the directed edge from v_i to v_j. Each edge has an associated travel cost c_{ij}^k when traversed by robot $r_k \in R$, $k = 1..m$. Elements of V are either passive sensors (pickup customers) or damaged sensors (delivery customers). Each customer v_i has a unitary demand q_i (-1 for pickup customers and 1 for delivery customers) and $\vec{q} = (q_1, q_2, \ldots, q_n)$. Node v_1 represents the base station with demand $q_1 = 0$, V_D is the set of delivery customers and V_P is the set of pickup customers. As in Sect. 4.2, the following statements also hold.

$$V = \{v_1\} \cup V_D \cup V_P$$

$$V_D = \{v_i \in V : q_i = 1\}$$

$$V_P = \{v_i \in V : q_i = -1\}$$

$$V_D \cap V_P = \emptyset$$

A unique type of commodity (sensors) is to be transported by the robots from one place to another. For simplicity, we will assume that all robots have the same maximum cargo capacity Q and initial cargo Q_0 ($0 \leq Q_0 \leq Q$). Not all graph nodes have to be visited. To ensure the service of all damaged sensors, a delivery node is assigned a profit p_i of 1. The profit p_i of a pickup node v_i and the base station v_1 is set to zero.

The goal is to find a minimal-cost feasible route plan $\Phi_{min} = (\vec{\varphi}_1, \vec{\varphi}_2, \ldots, \vec{\varphi}_z)$, $1 \leq z \leq m$, where $\vec{\varphi}_i$ is an individual robot route. A route plan Φ is regarded as *feasible* if all its routes are disjoint (i.e. have no nodes in common), each individual route is a Hamiltonian cycle originated at the base station, all damaged nodes are corporately replaced with passive nodes, each robot returns empty to the base station (i.e. no extra passive node is collected) and the robot's capacity constraint is never violated along any route. The *cost* of a route plan is defined here as the total distance traveled by all robots, its *cardinality* stands for the number of routes

it contains and its *length* refers to the total number of nodes in it, including the base station. More formally, for a given tuple T, the cardinality, length and cost of any feasible route plan Φ can be calculated as in (5.1), (5.2) and (5.3), respectively.

$$|\Phi| = \text{number of routes in } \Phi \tag{5.1}$$

$$len(\Phi) = 2 \cdot |V_D| - Q_0 + |\Phi| \tag{5.2}$$

$$f(\Phi) = \sum_{\vec{\varphi} \in \Phi} \sum_{e_{ij} \in \vec{\varphi}} c_{ij} \tag{5.3}$$

with c_{ij} being the Euclidean distance between v_i and v_j.

Decision variables are defined as follows:

- $x_{ij}^k = 1$ if edge e_{ij} has been visited by robot k, else 0
- $y_i^k = 1$ if node v_i has been visited by robot k, else 0
- l_{ij}^k = load of the k-th robot traveling along edge e_{ij}

This leads to the following integer linear programming formulation:

$$Min \sum_{k=1}^{R} \sum_{e_{ij} \in E} c_{ij}^k \cdot x_{ij}^k$$

subject to

$$\sum_{k=1}^{m} \sum_{v_j \in V \setminus \{v_i\}} x_{ij}^k \leq 1 \qquad \forall v_i \in V \setminus \{v_1\} \tag{5.4}$$

$$\sum_{k=1}^{m} \sum_{v_i \in V \setminus \{v_j\}} x_{ij}^k \leq 1 \qquad \forall v_j \in V \setminus \{v_1\} \tag{5.5}$$

$$\sum_{v_j \in V} x_{1j}^k = 1 \qquad \forall r_k \in R \tag{5.6}$$

$$\sum_{v_i \in V} x_{i1}^k = 1 \qquad \forall r_k \in R \tag{5.7}$$

$$\sum_{v_i \in V} x_{ij}^k - \sum_{v_i \in V} x_{ji}^k = 0 \qquad \forall v_j \in V \setminus \{v_1\},\ \forall r_k \in R \tag{5.8}$$

$$\sum_{v_j \in V \setminus \{v_1\}} l_{1j}^k = Q_0 \qquad \forall\ r_k \in R \tag{5.9}$$

$$\sum_{v_j \in V \setminus \{v_i\}} l_{ji}^k - \sum_{v_j \in V \setminus \{v_i\}} l_{ij}^k = q_i y_i^k \qquad \forall v_i \in V \setminus \{v_1\},\ \forall r_k \in R \tag{5.10}$$

$$0 \leq l_{ij}^k \leq Q x_{ij}^k \qquad \forall e_{ij} \in E,\ \forall r_k \in R \tag{5.11}$$

$$\sum_{k=1}^{m} \sum_{v_i \in V} p_i y_i^k = p_{min} \tag{5.12}$$

$$\sum_{v_i \in V} p_i y_i^k > 0 \qquad \forall r_k \in R \tag{5.13}$$

$$\sum_{v_i \in V} l_{i1}^k = 0 \qquad \forall r_k \in R \tag{5.14}$$

$$l_{ij}^k \text{ is integer} \qquad \forall e_{ij} \in E, \forall r_k \in R \tag{5.15}$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall e_{ij} \in E, \forall r_k \in R \tag{5.16}$$

$$y_i^k \in \{0, 1\} \qquad \forall v_i \in V, \forall r_k \in R \tag{5.17}$$

The objective function minimizes total costs, expressed in terms of total distance traveled by all robots to fix all damaged nodes. Constraints (5.4) and (5.5) ensure that each node except the base station is visited at most once. Equalities (5.6) and (5.7) guarantee that each robot departs from and returns to the base station. This does not imply that all robots will be used

to construct the route plan, for a robot r_k may remain at the base station by solely choosing the edge $x^k_{11} = 1$. Preservation of the flow per individual route is achieved through (5.8). Notice that the presence of (5.8) renders either (5.4) or (5.5) redundant, but we have decided to include both for the sake of clarity. Each robot leaves v_1 with an initial load Q_0 as stated in (5.9). Expressions (5.10) and (5.11) represent a robot's loading constraints. The difference in load of a robot entering and leaving a node must equal the demand of that node. A robot can never carry more than its maximum capacity. Constraint (5.12) ensures that all damaged units are visited by the robotic fleet. The total profit collected in the route plan is set to a minimum value p_{min}, equal to the number of delivery nodes $|V_D|$. Since only delivery customers have a profit of 1, these nodes are automatically included in the solution. Each individual route must repair at least one damaged sensor, as constraint (5.13) indicates. Every robot must return empty to the base station according to (5.14), which means that all passive sensors picked in the route have been dropped at the coordinates of the damaged nodes. Finally, constraints (5.15), (5.16) and (5.17) define the domain of the decision variables.

5.3 HSFA: A Hybrid Metaheuristic Approach for MC^{2}R

Given the intrinsic complexity of solving a MC^{2}R instance and the usually short time allotted to come up with a feasible route plan of good quality (due to the tight responsiveness constraints imposed on the WSN), a harmony-seeking firefly algorithm (HSFA) is put forth. It can be envisioned as a firefly swarm $\mathcal{S}$ of size M in which each member follows the exploratory principles featured by Harmony Search (HS). Foundational aspects of the Firefly Algorithm and Harmony Search were outlined in Sect. 2.1.2 and 2.1.2, respectively.

The Harmony Memory (HM) is a pivotal component of the HS approach. It is typically thought of as a centralized data structure (table) containing HMS elitist entries and that gets periodically updated by the current harmony (solution) under consideration. In our proposed implementation however, the HM resides collectively in the swarm (i.e. HMS = M) rather

than in a separate data structure in memory. It stores the personal-best position of every firefly and is dynamically updated by the fireflies themselves as their personal bests improve over time. The HSFA pseudocode is outlined in Alg. 6.

Since browsing across the feasible solution space alone is time-consuming (owing to the number of capacity violations and imbalanced number of customer types that could emerge in individual routes), we conduct the search starting from very poor, infeasible solutions and gradually drive them into feasibility under the influence of a weak Pareto dominance relationship, as described in Sect. 5.3.3. This phase consumes around 98% of the available CPU time (which usually varies between $5 \sim 600$ sec). The remaining 2% is set aside to construct a feasible route plan out of the best-performing infeasible solution found thus far. This is done by the repair heuristic in Sect. 5.3.4, which can indeed strike to some extent on the overall quality of the best infeasible candidate in order to return a full-blown, feasible solution when the CPU time elapses. In practice, this degradation of the final solution's performance grows with the network size (number of WSN nodes) yet can be conveniently controlled by reducing the sharpness of those repair steps that provoke an abrupt decline in the solution's quality.

5.3.1 Problem Descriptors and Algorithmic Parameters

Tables 5.1 and 5.2 display the MC^{2}R problem descriptors and HSFA parameters, respectively. While the former have their values dictated by WSRN operational constraints, proper values for the latter are discussed in Sect. 5.4. The initial cargo Q_0 was set to zero in order to allow for a wider exploration of the infeasible search space, as the robots have to collect all the necessary pickups from the field.

5.3.2 Solution Encoding

A *cyclic path representation* is used to encode the route plan for the robotic team in the firefly's position, e.g. $\vec{X}_i = (4, 5, 1, 6, 2, 7, 1, 3)$ means that two routes, viz 1-6-2-7-1 and 1-3-4-5-1, originate and conclude at the base station v_1. The length (number of components) of

Algorithm 6 HSFA for MC^{2}R problem instances

Input: M, Q, R, CPU, HMAR, PAR, pElite, p_R
Output: Best-ever route plan found $\vec{X}_g$

initializeDataStructures( );
for i = 1 to M **do** ▷ initialize swarm
    $\vec{X}_i \leftarrow$ generateNewSolution( );
end for
updatePickupUsage( );
while tic( ) ≤ 98% CPU **do**
    $\vec{X}_g \leftarrow$ EvaluateSwarmBrightness( );
    for i = 1 to M **do**
        $\vec{X}'_i \leftarrow \vec{X}_i$;
        if rand( ) ≤ HMAR **then** ▷ $\vec{X}_i$ draws from HM
            follower ← false;
            for j = 1 to M **do**
                if $\vec{X}_j \preceq \vec{X}_i$ **then**
                    follower ← true;
                    $\vec{X}^*_j \leftarrow$ chooseInformer(pElite);
                    $\beta \leftarrow$ attractiveness($I_0(\vec{X}_i), I_0(\vec{X}^*_j)$);
                    $\vec{X}'_i \leftarrow$ copyFromInformer($\vec{X}^*_j, \vec{X}'_i, \beta$);
                    if rand( ) ≤ PAR **then**
                        $\vec{X}'_i \leftarrow$ perturbSolution($\vec{X}'_i$);
                    end if
                    updateHM($\vec{X}'_i$);
                end if
            end for ▷ j loop
            if not follower **then**
                $\vec{X}'_i \leftarrow$ perturbSolution($\vec{X}'_i$);
                updateHM($\vec{X}'_i$);
            end if
        else ▷ improvise a new firefly
            $\vec{X}'_i \leftarrow$ generateNewSolution( );
            updateHM($\vec{X}'_i$);
        end if
        $\vec{X}_i \leftarrow \vec{X}'_i$;
    end for ▷ i loop
    updatePickupUsage( );
    updateParameters( );
end while
$\vec{X}_g \leftarrow$ repairBestSolution($\vec{X}_g$);
return $\vec{X}_g$;

Table 5.1: MC^{2}R Problem Descriptors

Symbol	Sampling Range	Description
Q	1 - 5	Maximum robot cargo capacity.
m	2 - 5	Number of robots in the team.
NS	30 - 500	Total number of sensor nodes.
DN	5% - 25%	Percentage of damaged nodes.
maxCPUTime	5 - 600 sec	Maximum time available for computations.

Table 5.2: HSFA Parameters

Symbol	Sampling Range	Description
M	20 - 60	Swarm size (number of fireflies).
HMAR	[0;1]	HM Acceptance Rate. Probability with which a firefly will draw past good solutions from the HM.
PAR	[0;1]	Pitch Adjustment Rate. Probability with which a firefly will perturb the solution drawn from the HM.
pElite	[0;1]	Probability with which a firefly will select its personal-best solution as informer for another firefly. Otherwise, it will select its current position as informer.
p_R	20% - 80%	Percentage of pickup nodes in a firefly's position that will be replaced with unvisited pickups.

firefly $\vec{X_i}$'s position will vary from one swarm member to another, being $2 \cdot |V_D| + |\vec{X_i}|$, where $|\vec{X_i}|$ is the number of routes encoded in the route plan $\vec{X_i}$ (i.e. number of 1's in the vector) as defined in (5.1) and it ranges from 1 to m. Hence, we are in presence of a variable-length firefly swarm.

5.3.3 Navigation across the Infeasible Search Space

The brightness (light intensity) of a firefly $\vec{X_i}$ is defined by (5.18)

$$I_0(\vec{X_i}) = (f_1(\vec{X_i}), f_2(\vec{X_i}), f_3(\vec{X_i}), f_4(\vec{X_i})) \tag{5.18}$$

where $f_1(\cdot)$ is the number of robots in the route plan, $f_2(\cdot)$ the total distance traveled, $f_3(\cdot)$ the total imbalance factor (for a route, it is the absolute value of the load with which the robot

arrives at the base station) and $f_4(\cdot)$ the number of capacity violations. The first objective aims at minimizing the number of robotic units engaged in the current sensor relocation task. This is desirable since the "idle" robots can replenish their batteries at the base station and be dispatched in case of emergency to a conflictive region in the deployment field. Objective 2 is improved as shorter route plans are discovered, which means that the network repair latency is reduced too. The last two objectives capture the infeasibility permeating a route plan $\vec{X}_i$. As the algorithm progresses, the joint minimization of the four objectives is pursued.

It is said that firefly $\vec{X}_j$ is brighter than $\vec{X}_i$ (and denoted by $\vec{X}_j \preceq \vec{X}_i$) if $\vec{X}_j$ *weakly dominates* $\vec{X}_i$, i.e. $\vec{X}_j$ is no worse than $\vec{X}_i$ across all objectives ($f_k(\vec{X}_j) \leq f_k(\vec{X}_i)\ \forall k = 1..4$) and at least one objective is improved ($\exists k \mid f_k(\vec{X}_j) < f_k(\vec{X}_i), k = 1..4$). If that happens, $\vec{X}_j$ becomes the attractor (or *informer*) for $\vec{X}_i$, which is referred to as the *follower* and will try to borrow some good components from $\vec{X}_j$.

The procedure in line 1 computes the distance matrix $\mathcal{D}(\cdot,\ \cdot)$ between any pair of graph vertices and sets the pickup usage vector $\mathcal{P}(\cdot)$ to $\vec{0}$. The former will aid in calculating the "potential" of a pickup node (i.e. its nearness to the set of reported delivery nodes) whereas the latter will play an instrumental role in selecting those pickup units that have not been quite represented in previous firefly swarms, thus fostering the exploration of neglected pickups and their inclusion in the group of route plans encoded by the swarm members at the current iteration. The `initializeDataStructures( )` method in line 1 also computes the potential vector $\vec{\varphi}$, where $\varphi_j = 1/\sum_{v_k \in V_D} \mathcal{D}(v_j, v_k),\ j = 1..|V_P|$. This static vector gives us an idea of how desirable it is to visit a particular passive unit for sensor replacement in terms of its geographical location being close to the group of damaged nodes.

The entire firefly swarm is initialized in lines 2–4. For each individual, it is first randomly decided a number of routes $1 \leq r \leq R$, then r 1's and all delivery nodes will be added and the vector shuffled. The remaining components are filled with pickups by the roulette wheel selection rule shown in (5.19).

$$p_j = \frac{\varphi_j/(\mathcal{P}_j + 1)}{\sum_{k \in V_P} [\varphi_k/(\mathcal{P}_k + 1)]} \tag{5.19}$$

where p_j is the probability that the pickup node v_j will be chosen, $\varphi_j = 1/\sum_{v_k \in V_D} \mathcal{D}(v_j, v_k)$ its "potential" and $\mathcal{P}_j$ its usage counter value. Lines 5 and 34 update the $\mathcal{P}$ vector by adding to the present counter for each v_j the number of times it appears in the encoding of all members of the current swarm. The idea behind the selection rule is to promote the inclusion of those pickups that have been least used in earlier firefly swarms and exhibit good potential, i.e. their geographical location lies fairly close to many delivery units.

Line 7 calculates the brightness of every individual and initializes/updates the best firefly $\vec{X}_g$ with an arbitrary non-dominated solution in the Pareto front. The tic( ) function checks the current clock time and rand( ) generates a number $\sim$ U(0, 1). In line 15, $\vec{X}_j$ will recommend its personal-best position (stored in the HM) as informer for $\vec{X}_i$ with probability pElite or its current encoding otherwise.

The strength of the attractiveness β in line 16 between follower $\vec{X}_i$ and its informer $\vec{X}_j^*$ is given by the average improvement (in %) of the informer over the follower across all improving objectives. The method in line 17 copies from the informer to $\vec{X}_i{}'$ exactly β% consecutive vector components, starting at an arbitrary location and as long as the copy does not remove any delivery unit in the follower and achieves a reduction in distance.

This HM-partially-borrowed solution is probabilistically perturbed in line 19 by randomly applying one of the following three local search operators:

- *Delivery Exchange*: Swaps two random delivery nodes in two arbitrarily selected routes.
- *Pickup Exchange*: Behaves idem to *Delivery Exchange* but with the pickup nodes.
- *Pickup Replacement*: Substitutes p_R% of the pickups in the firefly encoding with more profitable, unvisited ones as per the rule in (5.19)

Notice that the application of the first two operators may not necessarily improve the can-

didate solution $\vec{X}_i'$ yet the last operator only modifies it if a better individual (i.e. one with shorter total distance) was obtained.

The `updateHM( )` method in lines 21, 26 and 30 replaces $\vec{X}_{i-pbest}$ (i.e. the personal best of $\vec{X}_i$ in the HM) with $\vec{X}_i'$ if $I_0(\vec{X}_i') \succ I_0(\vec{X}_{i-pbest})$ and updates $\vec{X}_g$ accordingly.

The `updateParameters( )` method dynamically modifies the parameter values as explained in Sect. 5.4.

Finally, $\vec{X}_g$ will be turned into a feasible solution in line 36 should signs of infeasibility still persist as explained next.

5.3.4 Building a Feasible Final Route Plan

Our repair heuristic takes the route plan $\mathcal{R}$ encoded by $\vec{X}_g$ and builds a feasible final solution if needed. Since the navigation across the infeasible space ensures that all the delivery nodes will be present in $\mathcal{R}$ as well as the necessary number of pickups, the repair heuristic will focus on balancing the number of deliveries $|V_D|_R$ and pickups $|V_P|_R$ for each route $R \in \mathcal{R}$. Afterwards, it will rearrange the nodes in each route in order to remove any capacity violation. Notice that the first operation may involve multiple routes whereas the second one may apply to each individual route.

(1) *Balance routes*: See Algorithm 7 below.

(2) *Eliminate capacity violations*: For each route $R \in \mathcal{R}$ and starting from v_1, find the location y where the first capacity violation occurs. Then swap the node at y with a node at $z > y$ (cyclically) that removes the violation and optimizes $f_2(\cdot)$. Do the same for each detected capacity violation from this point onwards.

5.4 HSFA Experimental Study

Since MC^{2}R is a brand new problem first solved by HSFA, there is no other competing scheme yet that serves as benchmark. However, we can validate our results against those

Algorithm 7 Balancing routes in infeasible route plan

Input: route plan $\mathcal{R}$ with possibly imbalanced routes
Output: route plan with balanced routes $\mathcal{R}'$

$\mathcal{R}' \leftarrow \mathcal{R}$;
$\delta(R) \leftarrow \sum_{v \in R} q(v), \forall\, R \in \mathcal{R}'$; ▷ imbalance factor
$\lambda(R) \leftarrow \text{abs}(\delta(R))/|R|, \forall\, R \in \mathcal{R}'$; ▷ disturbance index
$S \leftarrow \{R \in \mathcal{R}' : \delta(R) \neq 0\}$; ▷ set of imbalanced routes
while $S \neq \emptyset$ **do**
    $R^* \leftarrow \underset{R \in \mathcal{R}'}{\text{argmin}}\ \lambda(R)$;
    if $\delta(R^*) > 0$ **then** ▷ add pickups
        sort the pickups in $S \setminus R^*$ in decreasing order of
        their potential $\varphi(\cdot)$ w.r.t. $V_D(R^*)$.
        transfer $p_1^*, p_2^*, \ldots, p_{\delta(R^*)}^*$ to R^*; update $S \setminus R^*$
    else ▷ add deliveries
        sort the deliveries in $S \setminus R^*$ in decreasing order of
        their potential $\varphi(\cdot)$ w.r.t. $V_P(R^*)$.
        transfer $d_1^*, d_2^*, \ldots, p_{-\delta(R^*)}^*$ to R^*; update $S \setminus R^*$
    end if
    recalculate $\delta(R)$, $\lambda(R)$ $\forall R \in S$
    $S \leftarrow \{R \in S : \delta(R) \neq 0\}$
end while
return $\mathcal{R}'$;

produced by the NN heuristic. Additionally, we will shed light on HSFA's exploration ability and the impact of the application of the repair heuristic over the best infeasible solution found.

5.4.1 Experimental Setup

All simulations were conducted in MATLAB R2009b on an Intel Core i7 CPU 860 @ 2.80 GHz with 6 GB of RAM under Windows 7 Home Premium with a 64-bit architecture. Synthetic data scenarios were generated, each holding information about the *problem descriptors* (WSRN configuration) and the *algorithm's parameters*. Their values have been uniformly drawn from the sampling ranges portrayed in Tables 5.1 and 5.2, respectively.

Once the problem descriptors are known, sensors are spatially distributed in a virtual field of $1,000 \times 1,000$ units. First, the depot's location δ (where the robot team lies) is chosen either deterministically at the field center $(0, 0)$, or randomly following a bivariate uniform distribution U, or stochastically following a bivariate normal distribution $\mathcal{N}(r, \sigma^2)$ with r being a uniform random 2D point. Second, the coordinates d_i of the delivery nodes are set either randomly following U, or stochastically centered around the depot after $\mathcal{N}(\delta, \sigma^2)$, or stochastically centered around the random point r following $\mathcal{N}(r, \sigma^2)$. Finally, the pickup nodes are positioned either randomly after U, or stochastically following $\mathcal{N}(d_i, \sigma^2)$ and surrounding the coordinates d_i of an arbitrary damaged sensor, or stochastically centered around the depot through $\mathcal{N}(\delta, \sigma^2)$, or stochastically centered around r via $\mathcal{N}(r, \sigma^2)$. The nine different spatial distributions are summarized in Table 5.3. This is the same approach used for SC^{2}R data scenario generation with GA-ACO (see Sect. 4.7.1).

For the simulations, synthetic MC^{2}R data scenarios have been created by randomly drawing values for the problem descriptors from their sampling ranges in Table 5.1 and by choosing an arbitrary spatial distribution from Table 5.3.

As to HSFA's parameters, the value of $M = 30$ was chosen after a careful empirical examination of the results obtained with $M \in \{20, 30, 40, 50, 60\}$. The remaining four parameters underwent an iteration-wise tuning procedure according to the following linear rules:

Table 5.3: WSN Spatial Distributions

Distribution	Depot δ	Delivery Node d_i	Pickup Node p_i
1	$(0, 0)$	U	U
2	$(0, 0)$	U	$\mathcal{N}(d_i, \sigma^2)$
3	$(0, 0)$	$\mathcal{N}(\delta, \sigma^2)$	$\mathcal{N}(\delta, \sigma^2)$
4	$(0, 0)$	$\mathcal{N}(\delta, \sigma^2)$	$\mathcal{N}(d_i, \sigma^2)$
5	U	U	U
6	U	U	$\mathcal{N}(d_i, \sigma^2)$
7	U	$\mathcal{N}(\delta, \sigma^2)$	$\mathcal{N}(\delta, \sigma^2)$
8	U	$\mathcal{N}(\delta, \sigma^2)$	$\mathcal{N}(d_i, \sigma^2)$
9	$\mathcal{N}(r, \sigma^2)$	$\mathcal{N}(r, \sigma^2)$	$\mathcal{N}(r, \sigma^2)$

$$\text{HMAR} = \text{HMAR}_{min} + (\text{HMAR}_{max} - \text{HMAR}_{min}) \times \frac{\text{tic}(\,)}{\text{CPU}} \tag{5.20}$$

$$\text{PAR} = \text{PAR}_{min} + (\text{PAR}_{max} - \text{PAR}_{min}) \times \frac{\text{CPU} - \text{tic}(\,)}{\text{CPU}} \tag{5.21}$$

$$\text{pElite} = \text{pElite}_{min} + (\text{pElite}_{max} - \text{pElite}_{min}) \times \frac{\text{tic}(\,)}{\text{CPU}} \tag{5.22}$$

$$\text{p}_R = \text{p}_{R_{min}} + (\text{p}_{R_{max}} - \text{p}_{R_{min}}) \times \frac{\text{CPU} - \text{tic}(\,)}{\text{CPU}} \tag{5.23}$$

where $\text{HMAR}_{min} = 0.3$, $\text{HMAR}_{max} = 0.8$, $\text{PAR}_{min} = 0.3$, $\text{PAR}_{max} = 0.9$, $\text{pElite}_{min} = 0.3$, $\text{pElite}_{max} = 0.9$, $\text{p}_{R_{min}} = 0.3$ and $\text{p}_{R_{max}} = 0.6$. These values have been experimentally determined by trial and error. HSFA's parameter values are modified at the end of each iteration so as to boost exploration at the outset of the algorithm and intensify exploitation towards the end of the allotted CPU time.

5.4.2 Simulation Results

The best route plan found by HSFA in 10 independent runs per data scenario is depicted in Table 5.4 as well as the final feasible solution returned by the repair heuristic.

Table 5.4: Route plans computed by HSFA and NN

Problem Instance Descriptors							Avg Best Infeasible Route Plan				Final Route Plan		Best-Final	NN	NN-Final
ID	NS	DN	R	Q	CPU	SD	$f_1(\cdot)$	$f_2(\cdot)$	$f_3(\cdot)$	$f_4(\cdot)$	$f_1(\cdot)$	$f_2(\cdot)$	(%)	$f_2(\cdot)$	(%)
1	36	5	2	3	364	4	1	1892.9	0	0	1	1665.752	12	2136.27	22
2	49	9	4	3	358	6	1	3255.03	0	0	1	2994.6276	8	3457.5	13
3	55	12	3	3	7	4	1	2962.41	0	0	1	2843.9136	4	3093.35	8
4	55	13	4	2	373	8	1	2312.62	0	0	1	1919.4746	17	2518.72	24
5	57	3	4	4	283	5	1	2401.21	0	0	1	1944.9801	19	2755.54	29
6	63	11	2	3	169	3	1	2252.81	0	0	1	2027.529	10	2660.79	24
7	69	12	2	3	203	9	1	2409.72	0	0	1	2337.4284	3	2614.23	11
8	74	7	5	5	395	4	1	2442.32	0	0	1	2100.3952	14	2731.87	23
9	82	13	2	2	87	4	1	1986.1	0	0	1	1906.656	4	2272.68	16
10	120	12	4	4	260	6	1	3753.65	0	0	1	3265.6755	13	3495.58	7
11	124	14	5	4	318	2	1	5252.28	0	0	1	5252.28	0	5252.28	0
12	134	25	4	2	7	7	1	3556.65	0	0	1	2916.453	18	3860.33	24
13	134	27	3	2	524	1	1	5925.41	0	0	1	4918.0903	17	6292.8	22
14	141	28	5	5	530	5	1	5306.25	0	0	1	4245	20	5714.48	26
15	162	37	3	2	31	2	1	7475.8	0	0	1	7475.8	0	7475.8	0
16	164	25	4	2	425	2	1	6112.29	0	0	1	5806.6755	5	6130.96	5
17	180	40	3	1	488	6	1	7552.52	0	0	1	7023.8436	7	8473.57	17
18	182	29	3	1	167	4	1	4706.79	0	0	1	4706.79	0	4706.79	0
19	196	31	2	2	356	1	1	6507.69	0	0	1	6182.3055	5	6907.86	11
20	202	44	4	4	23	9	1	4290.62	0	0	1	4290.62	0	4290.62	0
21	203	10	4	2	415	3	1	2045.84	0	0	1	2045.84	0	2045.84	0
22	207	29	4	3	122	8	1	3874.09	0	0	1	3719.1264	4	4048.28	8
23	215	34	5	3	519	2	1	7174.92	0	0	1	6313.9296	12	7289.25	13
24	220	24	4	2	79	6	1	6338.28	0	0	1	6338.28	0	6338.28	0
25	239	12	4	3	376	9	1	1620.28	0	0	1	1344.8324	17	1729.05	22
26	239	31	2	4	294	8	1	5281.77	0	0	1	5281.77	0	5281.77	0
27	241	14	4	3	69	5	1	3423.92	0	0	1	2841.8536	17	3787.95	25
28	248	60	2	2	240	8	1	5836.21	0	0	1	5310.9511	9	6065.86	12
29	256	44	2	2	227	7	1	3167.21	0	0	1	2660.4564	16	3560.78	25
30	256	59	3	3	21	2	1	8275.46	0	0	1	7365.1594	11	8632.84	15
31	259	34	4	5	590	9	1	4461.79	0	0	1	3881.7573	13	4898.9	21
32	288	32	5	4	472	2	1	6914.86	0	0	1	5531.888	20	6607.54	16
33	296	33	2	1	57	7	1	4892.85	0	0	1	4305.708	12	5907.09	27
34	312	28	5	4	418	7	1	3843.49	0	0	1	3843.49	0	3843.49	0
35	318	80	5	4	498	4	1	7521.36	0	0	1	6092.3016	19	8047.53	24
36	330	79	4	2	383	7	1	4459.59	0	0	1	4459.59	0	4459.59	0
37	343	48	5	3	447	2	1	7766.84	0	0	1	6990.156	10	7786.82	10
38	356	25	4	4	408	3	1	3871.81	0	0	1	3136.1661	19	4334.65	28
39	368	74	3	1	569	8	1	7088.25	0	0	1	5741.4825	19	7347.69	22
40	392	74	5	3	378	2	1	9864.3	0	0	1	9667.014	2	9900.05	2
41	413	95	2	5	381	2	1	12312.09	0	0	1	12312.09	0	12312.09	0
42	423	93	4	3	285	6	1	12399.25	0	0	1	10291.378	17	13173.04	22
43	433	56	2	3	136	2	1	8704.11	0	0	1	7398.4935	15	9122.5	19
44	433	87	5	4	393	4	1	9584.65	1	10	1	7955.2595	17	9221.51	14
45	447	27	4	5	136	2	1	5236.85	1	4	1	5027.376	4	5467.2	8
46	456	82	2	3	559	7	1	5832.2	1	5	1	6540.59	-12	7445.97	12
47	478	67	2	3	494	6	1	8735.81	0	12	1	7862.229	10	8994.24	13
48	481	82	4	2	198	2	1	11018.11	1	0	1	11695.937	-6	11055.78	-6
49	484	44	2	4	73	3	1	4296.67	1	9	1	3609.2028	16	5141.81	30
50	485	121	4	3	10	4	1	8135.21	1	3	1	7565.7453	7	8486.81	11
						AVERAGE	1	5526.66	0.12	0.86	1	5059.0869	8.46	5783.52	13.51

Notice how the total imbalance factor and number of capacity violations (objectives f_3 and f_4) are optimized in all cases save in a few large data scenarios ($NS > 400$). This speaks about the ability of the weak Pareto dominance relationship to gradually drive infeasible solutions into the feasible route plan space, even in networks with hundreds of nodes. This is also true for the total number of robots engaged in a sensor replacement operation (objective f_1), which is shrunk down to 1, in alignment with our desire to have as many idle robots as possible in the base station ready to respond to any ulterior emergency in the deployment field.

Another encouraging observation is the fact that the repair heuristic only degrades the best infeasible solution (in terms of objective f_2) by at most 12% and in only 2 of the 50 scenarios under discussion, given the good convergence of HSFA to feasible route plans. On the contrary, the repair step returns feasible route plans which are 8.46% shorter on average compared to their best infeasible counterparts and can achieve improvements of up to 20% of their total distance.

The last two columns of Table 5.4 contrast the final route plans yielded by HSFA with those obtained after applying the NN heuristic. To ensure a fair comparison baseline with HSFA, the number of routes in NN equals the number of robots allocated by HSFA in the final route plan. The results reveal that HSFA is able to find significantly better solutions in 90% of the data scenarios under consideration. This justifies the time spent in the calculation of the sensor replacement trajectories for the mobile robots, as the network repair latency and robot energy consumption are substantially minimized with HSFA's route plans.

5.5 Conclusions

We have targeted MC^{2}R, a novel problem of practical relevance for robot-assisted wireless sensor networks. A new generalization of the vehicle routing problem, here coined as VRP-SELPD, has been formulated and MC^{2}R modeled as a special case of a VRP-SELPD scenario. Its solution was subsequently sought via HSFA, a metaheuristic algorithm leaning upon the hybridization of artificial fireflies and harmony search. This is the first scheme in literature that tackles MC^{2}R scenarios. The conducted empirical analysis indicates that promising solutions can be achieved in a limited time span by the proposed hybrid method.

The next chapter is concerned with the identification of defective sensor nodes under the well-known distributed framework of *diagnosable systems*, a preliminary step in the execution of the CBFCF and CBCR protocols already presented in Chapters 3 to 5.

CHAPTER 6

SYSTEM-LEVEL FAULT DIAGNOSIS

Detection of hardware and software faults in parallel and distributed systems (with WSN being a paradigmatic illustration of such decentralized environments) remains a challenge nowadays. Precise and efficient identification of the defective networked components is both a pivotal aspect in the integral blueprint of a truly fault-reactive WSN and a sorely needed input for the carrier-based protocols described in Chapters 3 and 4.

Knowledge pertaining to fault identification in diagnosable systems, also called the *system-level fault diagnosis* problem, envelopes the reader in the present chapter. After explaining its fundamentals, two well-known test models (invalidation and comparison) are introduced. Next, we unfold two population-based metaheuristic algorithms aimed at quickly locating the actual set of defective nodes and emphasize on their architectural features. Swarms of artificial particles and fireflies constitute their main building blocks, respectively. The latter approach outdoes the former in terms of reliable prediction of the faulty units, owing to a stronger degree of social communication among the distributed search agents. Yet both metaheuristic schemes exhibit a high convergence rate and low memory consumption requirements.

6.1 Introduction

The emergence of innovative architectural and communication protocols has given rise to the next generation of decentralized systems such as wireless sensor and actuator networks [108] and cloud computing [141]. It is from the standpoint of these technological advancements that we witness a revival among the scientific community when it comes to fault identification protocols, as processing units in distributed and parallel systems are subject to both hardware and software faults. Since undetected faults lead to system errors with unpredictable results, efficiently diagnosing the system's status (i.e., identifying which nodes are faulty and which are fault-free) still remains a serious challenge for committed researchers.

One popular framework in literature for detecting damaged units is referred to as *system-level fault diagnosis*. In such scenarios, the process of spotting faults bears a distributed nature (as it is carried out by the system nodes themselves, which periodically test each other) whereas the final assignment of the nodes' statuses (either faulty or fault-free) exhibits some degree of centralization (which is indeed desirable for reliability purposes).

Given the underlying assumption that every system node is tested by other peers, a *test model* is then a set of rules for enunciating how such task can be carried out. Different test models [31] [80] [99] [120] have been proposed in literature. Two well-known testing frameworks are the invalidation and the comparison models. They are among the most studied topics in the field. Both models assume that each test outcome is of binary nature and indicates the status of the tested node. The system's status is hence derived from the ensemble of test results, which is known as the *input syndrome* and stands as the major fault identification vehicle.

Thorough examination of the input syndrome under an arbitrary network topology and guided by the set of rules enforced by each test model may imply a non-polynomial complexity. Furthermore, given the discrete nature of the node assignment process (every node is labeled as either faulty or fault-free), system-level fault diagnosis can be modeled as a combinatorial optimization problem, whose (optimal) solution is the set of node assignments that

entirely matches the system's input syndrome.

The next section reviews the characteristics of the connection assignments associated with the two test models under consideration. Then Sect. 6.3 and 6.5 unwind the swarm intelligence algorithms for system-level fault diagnosis that are put forward in this chapter.

6.2 Two Popular Test Models

Several models to identify faults in distributed systems appear in literature, as discussed in Chapter 2. In this section, we will focus on the popular *invalidation* and *comparison* models. Both assume that each entity is capable of testing a particular subset of the other entities in the system and every test has a binary character. A test outcome indicates whether the node is either *faulty* (1) or *fault-free* (0). Moreover, the group of all faulty units in a system is called the *fault set*. In order to narrow down the complexity of the fault diagnosis problem, researchers frequently work with *t-diagnosable* systems, in which the cardinality of any fault set is restricted to at most t units.

Although both diagnosis models portray the problem in a graph-like fashion and use the collection of test outcomes as a means to derive the status of every node, they differ in the graph representation and the implicit assumptions on the test results.

6.2.1 The PMC Model

In the invalidation or PMC model (named after its authors in [120]), the problem is represented as a directed graph $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$ being the set of entities (processors, sensors, etc.) and E the set of edges. Each edge $e_{ij} \in E$ stands for a test performed by node v_i upon node v_j and whose binary outcome is denoted as σ_{ij}. The set of all test outcomes is called a *syndrome* and is symbolized by σ. Since a syndrome σ is a physical manifestation of an underlying fault set F, we usually write σ_F.

According to the PMC model, the system has a stationary nature, i.e. the statuses of all

nodes do not change during the diagnosis phase. It is also assumed that tests carried out by fault-free nodes are always correct while those executed by faulty devices are unreliable, i.e. yield arbitrary results.

Let $v_i \in V$ be any node in G. Then the set of nodes tested by v_i is $\Gamma(v_i) = \{v_j \in V : e_{ij} \in E\}$ and $\sigma(v_i) = \{\sigma_{ij} \in \sigma : j \in \Gamma(v_i)\}$ is the subset of the syndrome σ containing the results of the tests realized by v_i. In a similar way, $\Gamma^{-1}(v_i) = \{v_j \in V : e_{ji} \in E\}$ is the set of v_i's testers and $\sigma^{-1}(v_i) = \{\sigma_{ji} \in \sigma : j \in \Gamma^{-1}(v_i)\}$ the outcomes of all tests carried out upon v_i.

Definition 6.1. *A fault set $F \subseteq V$ is consistent with the syndrome σ under the PMC model if $\forall\, e_{ij} \in E$, neither $\sigma_{ij} = 0$ where $v_i \in V - F$ and $v_j \in F$, nor $\sigma_{ij} = 1$ where $v_i, v_j \in V - F$, holds.*

The above definition states that only fault-free nodes always give correct results. This assumption will play a pivotal role later on during the generation of candidate syndromes as part of the quest for the true fault set.

6.2.2 Comparison Model

In the comparison model [119], nodes perform tests in a pairwise fashion, i.e. some pairs of units test each other. The comparison between the two test outcomes (match/mismatch) stands as the foundation for deriving the nodes' statuses. More formally, in a comparison-based scenario, an undirected graph $G = (V, E)$ is used to model the system, where V is the set of units and E the collection of edges. Now each edge e_{ij} has an associated weight $\sigma_{ij} = 0$ when the two test results carried out by units v_i and v_j are identical or $\sigma_{ij} = 1$ otherwise. As in the PMC model, the collection of all edge weights creates the syndrome σ.

The comparison model also regards the system as static, likewise PMC. Now the main assumption is that two fault-free units yield identical test outcomes whereas any couple of faulty and fault-free nodes will yield mismatching results. However divergent standpoints arise when it comes to the test comparison of two faulty nodes. The asymmetric version of the comparison model [99] considers that two damaged nodes will always produce a disagreement while in the symmetric version [31], both a match and a mismatch are likely choices.

Let $v_i \in V$ be any node in G. Then the set of neighbors of v_i is $\Gamma(v_i) = \{v_j \in V : e_{ij} \in E\}$ and $\sigma(v_i) = \{\sigma_{ij} \in \sigma : j \in \Gamma(v_i)\}$ is the subset of the syndrome σ containing the test agreements/disagreements involving v_i.

Definition 6.2. *A fault set $F \subseteq V$ is consistent with the syndrome σ under the symmetric comparison model if $\forall\ e_{ij} \in E$, neither $\sigma_{ij} = 0$ where $v_i \in V - F$ and $v_j \in F$ (or $v_i \in F$ and $v_j \in V - F$), nor $\sigma_{ij} = 1$ where $v_i, v_j \in V - F$, holds.*

Definition 6.3. *A fault set $F \subseteq V$ is consistent with the syndrome σ under the asymmetric comparison model if $\forall\ e_{ij} \in E$, neither $\sigma_{ij} = 0$ where $v_i \in V - F$ and $v_j \in F$ (or $v_i \in F$ and $v_j \in V - F$) or $v_i, v_j \in F$, nor $\sigma_{ij} = 1$ where $v_i, v_j \in V - F$, holds.*

6.2.3 Working with t-Diagnosable Systems

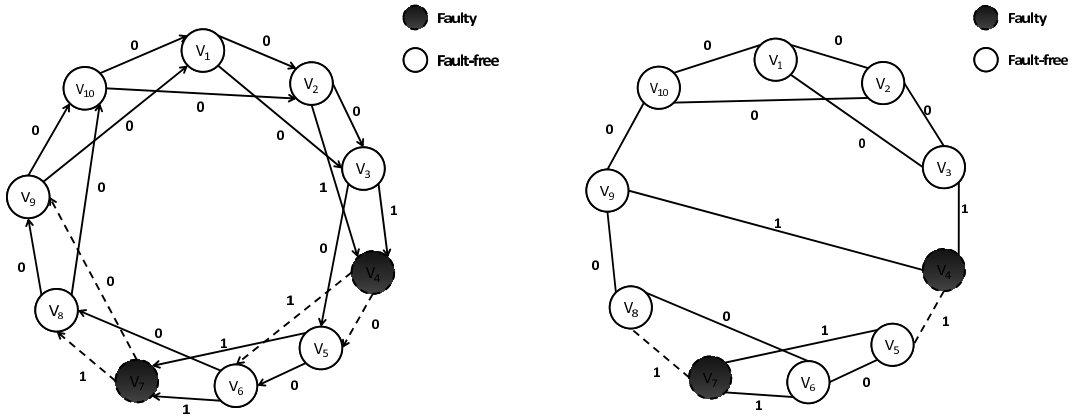
From the previous two subsections one may realize that, given an input syndrome σ which is compliant with the specifications of some model, multiple faults sets could possibly give rise to it. This makes system-level fault diagnosis a non-deterministic problem, which is of course very undesirable since the actual set of faulty units cannot be guessed with full certainty. To prevent this, we introduce the following definition.

Definition 6.4. *A system of n units is t-diagnosable iff the number of faulty units does not exceed t and for any consistent syndrome σ, there is only one fault set F that originated it.*

To ensure that F is unique for a given σ, the edges of the connection graph can be arranged in such a way that the system being modeled becomes t-diagnosable. In the sequel, we work with a type of graph called $G_t(n)$ which is easy to generate and known to be t-diagnosable.

Definition 6.5. *A system represented by a test graph $G = (V, E)$ with $n = |V|$ is a $G_t(n)$ design iff i) $n \geq 2t + 1$ and ii) each node is tested by at least t others.*

For the PMC model, it was proved in [65] that a $G_t(n)$ system in which no two units test each other is $t-$diagnosable. As to the comparison models, [119] concluded that the



(a) Directed edge labels represent test outcomes. Solid and dashed arrows stand for reliable and unreliable tests, respectively.

(b) Undirected edge labels represent the comparison result of pairwise test outcomes.

Figure 6.1: A $G_2(10)$ test assignment graph under the PMC (6.1(a)) and asymmetric comparison (6.1(b)) models.

asymmetric version is t-diagnosable if every unit is linked to at least $t + 1$ other units whereas in the symmetric version there must be at least $2t$ links to other units per node. Figure 6.1 displays a $G_t(n)$ graph under the PMC and asymmetric comparison models, respectively.

Finally, let us point out that, irrespective of the diagnosis model being used, the problem of fault detection in distributed and parallel systems can be posed as a combinatorial optimization problem if we undertake a quest, over the discrete space of all possible fault sets F, of the actual fault set $\overline{F}$ which univocally generates the known input syndrome $\sigma_{\overline{F}}$. This task has NP-hard complexity for systems with arbitrary topologies [17] and calls for the application of heuristic methods to informedly navigate over the search space until $\overline{F}$ is found. The next section explains how fault diagnosis can be efficiently tackled with a popular bio-inspired metaheuristic approach.

6.3 Detecting Faults with Particle Swarm Optimization

We model system-level fault diagnosis as a combinatorial search problem undertaken by a binary PSO approach (BPSO-FD). Recall the basics of the binary PSO formulation in 2.1.2.

The goal is to find, among many possible fault sets, the true ensemble $\overline{F}$ that entirely matches the collection of test outcomes contained in the input syndrome $\sigma_{\overline{F}}$. In our solution, the diagnosis process needs to run in a dedicated node.

6.3.1 Particle Encoding and Initialization

In BPSO-FD, each particle stands for a candidate fault set F. Therefore, we encode its position $\vec{X}_i$ as a binary vector whose dimensionality $d = |V|$ is given by the system size (number of distributed or parallel units). The status of the j-th node ($j \in \{1..d\}$) will be denoted by $X_{ij} = 1$ if we guess that v_j is faulty or $X_{ij} = 0$ otherwise.

At the algorithm outset, all particle velocities are randomly initialized with real values in $[0, 1]$ whereas for each particle position, it is first decided at random the number of faulty units $nf \sim U(1, t)$ the particle will encode and then nf bits will be arbitrarily set to one. By doing so, we are trying to promote diversity in the initial swarm, since the cardinality of its encoded fault sets will vary stochastically within the feasible bounds imposed by a t-diagnosable system.

For the potential fault set F represented by each particle, we generate its corresponding candidate syndrome σ_F before the particle's fitness can be calculated.

6.3.2 Fitness Function

The fitness (quality) of any particle is measured as the degree of resemblance between the input syndrome $\sigma_{\overline{F}}$ and the candidate syndrome σ_F of its associated fault set F. The computation of the fitness function is strongly model-dependent in spite of the common ground shared by all models.

Candidate Syndrome Generation

Of all test models considered in this study, only the asymmetric comparison model is entirely deterministic. That is, given the assumption of which nodes are faulty and which are fault-free (particle encoding), we follow the asymmetric model rules stated in Section 6.2.2

and assign a label to every edge e_{ij} in the test assignment graph G. The collection of these labeled edges constitutes our candidate syndrome σ_F.

Since the invalidation and symmetric comparison models have a non-deterministic nature concerning the tests carried out by faulty nodes (in the former case) and the match/mismatch between a pair of damaged devices (in the latter case), one can make the reasonable assumption that these edges in the candidate syndrome are equal to those in the input syndrome, i.e.

$$\sigma_F(v_i) = \sigma_{\overline{F}}(v_i) \quad \forall\, v_i \in F$$

Then we generate the remaining edge labels in σ_F in full compliance with the rules defined for each model, which were clearly portrayed in Section 6.2.2. We denote by $g : F_i \rightarrow \sigma_{F_i}$ the mapping from a candidate fault set F_i to its corresponding syndrome σ_{F_i}.

The Particle's Fitness

A particle is said to have better quality than others if the candidate syndrome associated with its encoding (fault set F) resembles the input syndrome more closely. From the local viewpoint of node v_i, this is equivalent to count how many labels of the incoming/outgoing edges coincide in both syndromes. For the comparison models, expression (6.1) models the node-level similarity.

$$f\left(\sigma_F, \sigma_{\overline{F}}, v_i\right) = \frac{|\sigma_F(v_i) \cap \sigma_{\overline{F}}(v_i)|}{|\Gamma(v_i)|} \tag{6.1}$$

For the PMC model, we are to take into account the dual nature of the nodes as a tester or tested. Equation (6.2) depicts the similarity between both syndromes from v_i's standpoint.

$$f\left(\sigma_F, \sigma_{\overline{F}}, v_i\right) = \frac{|\sigma_F(v_i) \cap \sigma_{\overline{F}}(v_i)| + |\sigma_F^{-1}(v_i) \cap \sigma_{\overline{F}}^{-1}(v_i)|}{|\Gamma(v_i)| + |\Gamma^{-1}(v_i)|} \tag{6.2}$$

Now we can define in (6.3) the fitness function for BPSO-FD, which is the overall similarity between both syndromes and is model-independent.

$$f(\sigma_F, \sigma_{\overline{F}}) = \frac{\sum_{v_i \in V} f\left(\sigma_F, \sigma_{\overline{F}}, v_i\right)}{|V|} \tag{6.3}$$

The above equation can be seen as the correctness probability of the potential fault set F being the actual fault set $\overline{F}$. Because we are working with t-diagnosable systems solely, then it is guaranteed that there is a unique fault set F that makes $\sigma_F = \sigma_{\overline{F}}$. Remark that (6.1) and (6.2) take values in [0, 1], which thus defines the image of (6.3) to be the same interval.

6.3.3 Position Update Rule

We have used (6.4), which was introduced in a recent study [152] and yielded enhanced experimental results for our problem, where rand( ) ~ U(0,1) and $k \in \{1..d\}$.

$$X_{ik} = \begin{cases} 1, & \text{if rand( )} \leq \dfrac{1}{1 + \exp(-1.5 \cdot d \cdot V_{ik})} \\ 0, & \text{otherwise} \end{cases} \tag{6.4}$$

6.3.4 Handling Infeasible Solutions

The application of the stochastic rule for the position update in (6.4) to every component of the particle's encoding can lead us to an infeasible solution, which is a certain configuration of node guesses in which there are more than t faulty units. Since this situation is unacceptable in t-diagnosable systems, we gracefully manage it in the following way:

(1) Compute the node-level similarity by either (6.1) or (6.2) depending on the underlying diagnosis model.

(2) If the particle encoding is the zero vector, generate a random number $b \in [1..t]$ and flip (turn to 1) the b bits with the lowest fitness values. Otherwise, let k be the number of 1's in the particle (notice that $k > t$). Then generate a random number $b \in [k - t..k - 1]$ and flip (turn to 0) the b bits set to 1 with the lowest fitness values.

The above procedure turns infeasible into feasible particles by flipping the bits (nodes) with lowest local resemblance to the input syndrome, in an attempt to accelerate the convergence of the algorithm towards the actual fault set.

6.3.5 Stop Criterion

BPSO-FD stops either when it reaches an upper bound of the allotted running time (expressed as a function of the number of iterations) or when it finds the actual fault set $\overline{F}$, i.e. a particle encoding F with fitness value $f(\sigma_F, \sigma_{\overline{F}}) = 1$. This fault set F is guaranteed to be unique in t-diagnosable systems (cf. Definition 6.4). If the time expires and $\overline{F}$ has not yet been found, the individual F with highest quality is returned.

6.3.6 The BPSO-FD Algorithm

Algorithm 8 unfolds the entire pseudo-code for the BPSO-FD scheme.

6.4 BPSO-FD Experimental Study

We have empirically tested the performance of BPSO-FD under t-diagnosable systems of various sizes for both invalidation and symmetric comparison models. Our algorithm has been contrasted to the one in [43, 146], which employs an artificial immune system (AIS) as the optimization approach, in terms of ϕ = total number of individuals (candidate fault sets) probed before identifying the true fault set and ρ = CPU time needed to find $\overline{F}$. The simulations were conducted on an Intel Core2 Duo E4500 2.2GHz with 2 GB of RAM. Both schemes were coded in Java using JDK 1.6 and tried with the $G_{24}(50)$ and $G_{49}(100)$ t-diagnosable graphs.

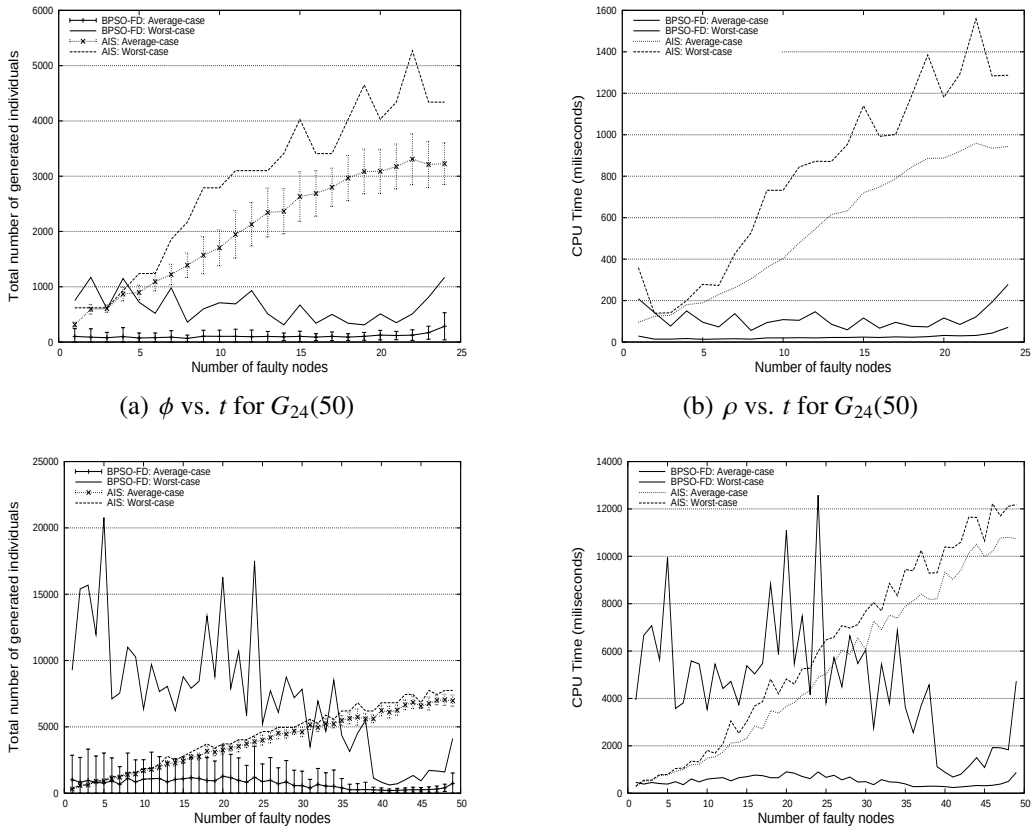
We used the AIS parameter configuration outlined in [146] whereas BPSO-FD's parameters were set as follows: $c_1 = c_2 = 2.05$. The particle neighborhood follows the "global best" topology with the use of constriction factor for the velocity update rule, as in [20].

For each graph, the number of faults x has been varied from 1 to t and for every value of

Algorithm 8 Binary PSO to Fault Diagnosis (BPSO-FD)

Input: syndrome $\sigma_{\overline{F}}$, functions $f(\cdot), g(\cdot)$, parameters c_1, c_2, M, maxIterations
Output: best fault set found $\vec{p_g}$

```
1:  iteration ← 0;
2:  compute constriction factor χ using (2.5);
3:  initialize swarm as shown in Section 6.3.1;
4:  repeat
5:      iteration ← iteration +1;
6:      for i = 1 to M do                                   ▷ for each particle X_i
7:          update particle velocity V_i using (2.3);
8:          update particle position X_i using (6.4);
9:          if X_i infeasible then
10:             apply feasibility scheme in Section 6.3.4;
11:         end if
12:     end for
13:     for i = 1 to M do                                   ▷ for each particle X_i
14:         σ_{F_i} ← g(X_i);                               ▷ generate candidate syndrome
15:         compute the fitness f(σ_{F_i}, σ_{F̄}) using (6.1) – (6.3);
16:         update local best p_i accordingly;
17:     end for
18:     update global best p_g accordingly;
19: until f(p_g, σ_{F̄}) = 1 or iteration >= maxIterations;
    return p_g;
```



(a) ϕ vs. t for $G_{24}(50)$

(b) ρ vs. t for $G_{24}(50)$

(c) ϕ vs. t for $G_{49}(100)$

(d) ρ vs. t for $G_{49}(100)$

Figure 6.2: Performance of BPSO-FD and AIS under the PMC model.

x, we have randomly generated 100 fault sets of cardinality x and their average results and standard deviations have been reported. In all cases, both BPSO-FD and AIS succeeded in discovering the actual fault set yet they exhibit remarkable performance differences.

6.4.1 Experiment 1: Performance under the PMC Model

Figure 6.2 displays the behavior of AIS and BPSO-FD under the invalidation model. As expected, the more faulty nodes to be detected, the larger the amount of computational resources consumed, i.e. number of explored individuals ϕ (particles or antibodies) and running time ρ. Nevertheless, we witness a nearly constant growth rate for BPSO-FD in both indicators for the $G_{24}(50)$ graph, as opposed to AIS which undergoes a polynomial growth pace (in some cases it is actually exponential, check Fig. 6.2(a) and 6.2(b)).

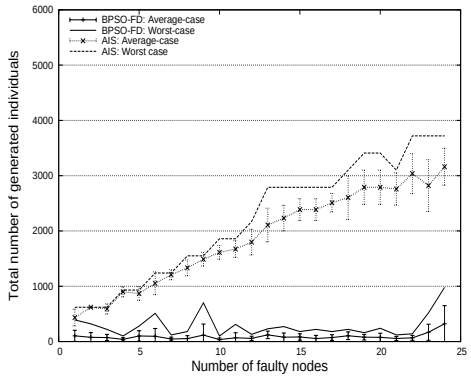
Furthermore, observe the huge gap between both schemes in terms of ϕ (42x wide in its peak) and ρ (5.2x longer in the worst scenario) in order to arrive at the actual fault set $\overline{F}$. The rapid convergence of the binary PSO method to the optimum is due to the presence of the constriction factor in the velocity update equation (2.3), which enforces a greater exploitation of the best solutions found by the swarm and also to the use of the global best topology, which is quite effective in unimodal optimization landscapes. We contrast this behavior with the erratic convergence rate portrayed by AIS, whose reliance upon cloning and mutation operators leads to a rather aimless exploration of the discrete space.

Regarding the $G_{49}(100)$ system, Fig. 6.2(c) and 6.2(d) reveal that BPSO-FD behaves in a disappointing fashion for the worst case with regards to both evaluation metrics in systems bearing up to about 30 faults, yet from there onwards it shows better performance than AIS. However, it is always far superior to its competitor in the average case under any fault set cardinality. In this medium-size system, AIS managed to reduce the breach between its average and worst-case runs, for stagnation is partially avoided by the larger search space due to the broader range of antibodies generated through cloning and mutation.

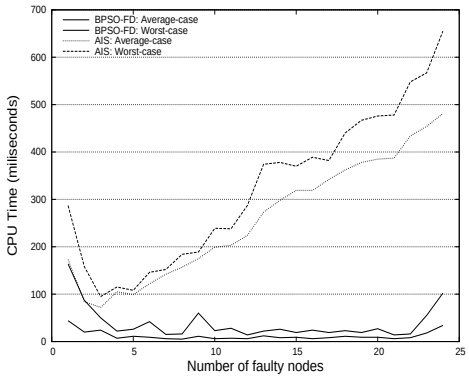
6.4.2 Experiment 2: Performance under the Comparison Model

The findings uncovered by Figure 6.3 for the symmetric comparison model are even more encouraging. For $G_{24}(50)$, the separation between the nature-inspired and the evolutionary representatives is more noticeable in Fig. 6.3(a), with short standard deviations of ϕ for BPSO-FD's average case. The higher degree of uncertainty among the population members in the AIS algorithm is evidenced by its fairly wide confidence intervals. The same behavior is repeated in terms of ρ for both approaches (Fig. 6.3(b) and 6.3(d), bars omitted for clarity)

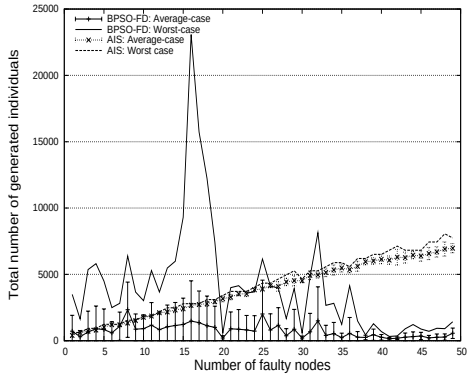
According to Fig. 6.3(c), the PSO-based scheme exhibits again an irregular tendency in the worst-case identification of the true fault set for the medium graph. We will deal with this flaw in more detail in Sect. 6.5. Yet observe that despite the rising number of damaged units to be spotted, its performance in the average case follows the quasi-constant pattern disclosed in



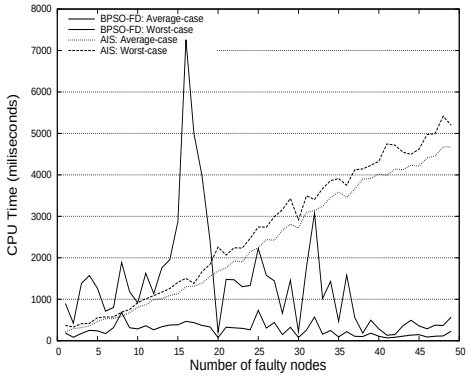
(a) ϕ vs. t for $G_{24}(50)$



(b) ρ vs. t for $G_{24}(50)$



(c) ϕ vs. t for $G_{49}(100)$



(d) ρ vs. t for $G_{49}(100)$

Figure 6.3: Performance of BPSO-FD and AIS under the symmetric comparison model.

the PMC model as well, which confirms our claim that BPSO-FD is a resilient fault diagnosis protocol for multiple test models.

6.5 Fault Identification with Binary Adaptive Fireflies

The suspicion that the worst-case irregular behavior of BPSO-FD might aggravate in larger systems (thus causing a poor and misleading identification of the defective devices) has triggered further research endeavors, which are described in this section.

From Figs. 6.3(c) and 6.3(d), one realizes that such faltering demeanor tends to vanish as more faulty nodes are believed to exist in the system (i.e. larger values of t). This is a tangible manifestation of the strong attraction exercised by the global-best particle over the remaining swarm members, which may drive their encoding into infeasibility. In presence of medium/large systems with low values of t, there are fewer feasible encodings that could be generated by the scheme in Sect. 6.3.4 to help particles escape the pitfall they were likely led into by the swarm's best agent. As $t \rightarrow |V|$, the infeasibility handling routine appears to boost its effectiveness.

In light of the above analytical insights, the intuitive solution to this problem seems to be diversifying the search even further by adding more "informers" to the velocity update rule of the search agents (particles). While single-informer PSO models are popular in literature and easy to implement, they can get us trapped in embarrassing situations like the aforementioned one. We witness a gradual shift in PSO research to multiple-informer architectures [35, 103], i.e. the velocity of each particle is sensitive to the position of more individuals in its neighborhood.

Another way of adding manifold informers to our discrete search engine is to escape the PSO orbit and land into a different algorithmic prototype. That is exactly the research avenue pursued in this study. We chose to lean upon the Firefly Algorithm (see Sect. 2.1.2) and its PSO-inspired movement rule to carry out system-level fault diagnosis. Our proposed method

(Firefly Algorithm to Fault Diagnosis, FAFD) completely eliminates the inadequate worst-case trend of PSO in medium-sized networks and its inability to swiftly capture the global optimum in large graphs, as will be shown in Sect. 6.6.

FAFD shares the same encoding as well as candidate generation and infeasibility management schemes and stop criterion already put forth in Sect. 6.3.1, 6.3.2, 6.3.4 and 6.3.5, respectively. It differs though in the manner in which fireflies update their positions.

6.5.1 Adaptive Light Absorption Coefficient

Equation (2.7) models the influence of the solutions attained by other fireflies over the current position of any individual. The attractiveness attenuation function is governed by two built-in parameters: β_0 and γ. The former describes the degree of attraction of two individuals colliding at the same location and is usually set to $\beta_0 \in [0, 1]$. The latter is the light absorption coefficient γ, which determines how increasing distance from the light emitter degrades its appeal to other individuals. All firefly solutions are equally interesting when $\gamma \to 0$ and completely irrelevant when $\gamma \to \infty$. A range that works well in practice is $\gamma \in [0, 10]$, as suggested by [150].

Superior results are possible when γ is problem-tailored. Instead of using a fixed value, we write γ in terms of the "characteristic length" of the subspace a firefly $\vec{X}_i$ is supposed to travel in. The adaptive version of γ is as follows:

$$\gamma(\vec{X}_i) = \frac{\gamma_0}{r_{max}(\vec{X}_i)} \tag{6.5}$$

where $\gamma_0 \in [0, 1]$ and $r_{max}(\vec{X}_i) = \max\{\text{dist}(\vec{X}_i, \vec{X}_j)\}$ for all $\vec{X}_j$ brighter than $\vec{X}_i$.

6.5.2 Binary Mapping Rule

The same kind of binary mapping applied to PSO in Sect. 6.3.3 is enforced here, but now utilizing the real-valued position vector $\vec{X}_i$ of an artificial firefly as input to the probabilistic

change rule in (6.6), which turns it into a binary-valued vector.

$$X_{ik} = \begin{cases} 1, & \text{if rand( )} \leq \dfrac{1}{1 + \exp(-1.5 \cdot d \cdot X_{ik})} \\ 0, & \text{otherwise} \end{cases} \tag{6.6}$$

6.5.3 The Pseudocode

Algorithm 9 unfolds the pseudocode of our method.

6.5.4 Time Complexity

An upper bound for the worst-case temporal complexity of FAFD is provided in this epigraph. The analysis of the spatial complexity is analogous and thus omitted for brevity.

Generation of the initial swarm (line 2) takes Mt time steps, as at most t bits per individual will be set to one. Since $t \leq (d-1)/2$ in a t-diagnosable system for PMC and symmetric comparison models, this is equivalent to Md. The loop in line 3 has 'maxIterations' as upper bound. The derivation of the candidate syndromes for the population in line 5 is the most expensive step, requiring Mdt steps with the PMC model and $Md(t+1)$ for the symmetric comparison model. In any case, such complexity is bounded by $O(Md^2)$.

Line 6 is carried out in Md steps whereas lines 7–8 are accomplished altogether in M steps. The calculation of $r_{max}(\vec{X}_i)$ has Md steps whereas the infeasibility handling scheme in line 32 needs $d \log d$ time slots for sorting the fitness values of all vector components plus d for flipping bits.

In total, the code snippet in lines 14–25 can be run in $2M^2d + 5M^2 - Md - M$ and that in lines 27–34 requires $Md \log d + 3Md + d$. This brings the overall FAFD time complexity to maxIter $\times$ $(Md^2 + M^2d + Md \log d)$. As simulations will reveal, both M and maxIter will take modest values in reality, so the expression boils down to $O(d^2 + d \log d)$.

Notice that the quadratic component above is not contributed by the meta-heuristic method itself (firefly optimization) but is due to the unfolding of the candidate syndromes. To the best

Algorithm 9 Firefly Algorithm to Fault Diagnosis (FAFD)

Input: syndrome $\sigma_{\overline{F}}$, functions $f(\cdot)$, $g(\cdot)$, parameters M, N, γ_0, α, β_0, maxIterations
Output: The best fault set found returned in $\vec{X}_{i^*}$

stop $\leftarrow$ false; iteration $\leftarrow$ 0;
initialize swarm S as shown in Sect. 6.3.1;
while not stop **do**
    iteration $\leftarrow$ iteration + 1;
    $\sigma_{F_i} \leftarrow g(\vec{X}_i); \forall\, i = 1..M$ ▷ generate candidate syndrome
    $I_0(\vec{X}_i) \leftarrow f(\sigma_{F_i}, \sigma_{\overline{F}});\ \ \forall\, i = 1..M$ ▷ compute fitness function
    $I_0^* \leftarrow \max\{I_0(\vec{X}_i)\}$; ▷ best fitness in this iteration
    $i^* \leftarrow \text{argmax}_i\{I_0(\vec{X}_i)\}$; ▷ index of best firefly

    if $I_0^* = 1$ || iteration > maxIterations **then**
        stop $\leftarrow$ true; **break**; ▷ termination criterion fulfilled
    end if

    for $i = 1$ to M **do** ▷ for each firefly $\vec{X}_i$
        $r_{max}(\vec{X}_i) \leftarrow \max_{\forall \vec{X}_j : I_0(\vec{X}_j) > I_0(\vec{X}_i)} \{\text{dist}(\vec{X}_i, \vec{X}_j)\}$;

        for $j = 1$ to M **do** ▷ for each firefly $\vec{X}_j$
            if $I_0(\vec{X}_j) > I_0(\vec{X}_i)$ **then** ▷ move $\vec{X}_i \rightarrow \vec{X}_j$
                $r_{ij} \leftarrow \text{dist}(\vec{X}_i, \vec{X}_j)$;
                $\gamma(\vec{X}_i) \leftarrow \gamma_0 / r_{max}(\vec{X}_i)$;
                $\beta(\vec{X}_i, \vec{X}_j) \leftarrow \beta_0 \cdot \exp(-\gamma(\vec{X}_i) \cdot r_{ij})$;
                apply movement rule in (2.8) $\forall k = 1..N$;
            end if
        end for
    end for

    move best firefly $\vec{X}_{i^*}$ according to (2.9) $\forall k = 1..N$;

    for $i = 1$ to M **do**
        $\vec{X}_i \leftarrow$ binary mapping rule in (6.6) $\ \forall k = 1..N$;
        if $\vec{X}_i$ is infeasible **then**
            apply infeasibility scheme in Sect. 6.3.4;
        end if
    end for
end while

of our knowledge, all fault identification algorithms dealing with t-diagnosable systems of arbitrary topologies run in $\Omega(d^2)$. The actual sought-after property is a rather steady average detection time as the number of faulty units in the system goes up. The next section confirms that FAFD truly fulfills this important goal.

6.6 FAFD Experimental Study

The same simulation setup outlined in Sect. 6.4 holds, except that a large graph $G_{249}(500)$ was brought in to test scalability. The average and worst-case executions of AIS, BPSO-FD and FAFD in the form of ϕ and ρ have been recorded. The 95% confidence intervals of the average case for both metrics are not shown in every plot for the sake of clarity. They are instead displayed mainly when highlighting differences between BPSO-FD and FAFD, provided either succeeded in discovering the true fault set $\overline{F}$.

6.6.1 Parameter Sensitivity Analysis

To guarantee a fair comparison baseline, BPSO-FD and FAFD have a common neighborhood topology ("global best") and the three protocols share the same population size ($M = 10$). Fig. 6.4 refers to a subset of all conducted trials that confirms this is the most realistic choice in terms of accuracy and convergence rate across quite a few configurations. In practice, a small number of agents manage to reasonably explore the binary N-dimensional space, which is good news for the application of these methods to resource-constrained environments.

The randomization factor in FAFD was set to $\alpha = 10^{-3}$, as it slightly outperforms other values. We need not too much randomness in the firefly movement rule, as the boundaries of the binary search space can be efficiently reached with the help of the solutions contributed by informer agents. The remaining FAFD parameters were set to $\gamma_0 = \beta_0 = 1$ after applying an identical trial-and-error rationale.

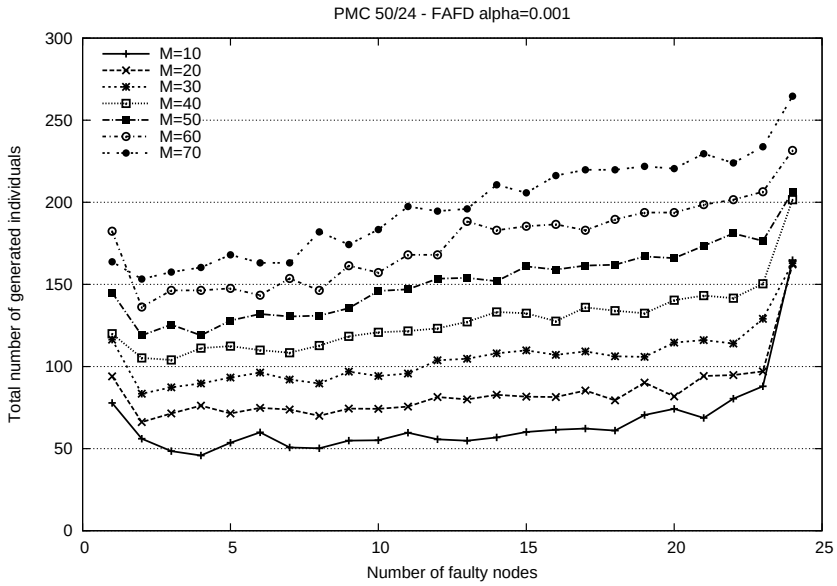
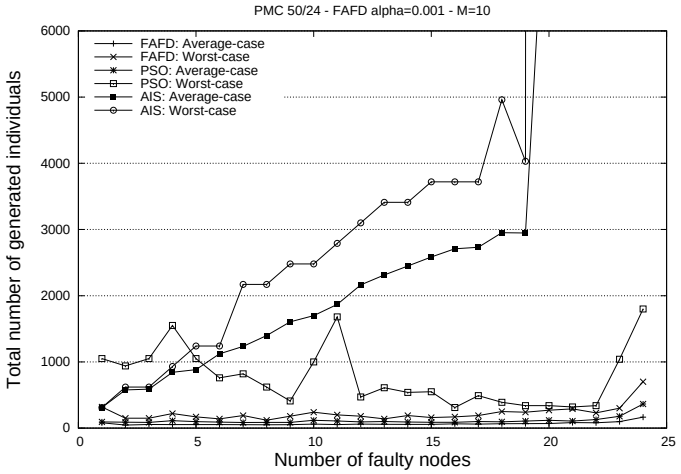


Figure 6.4: Swarm size selection for FAFD in a $G_{24}(50)$ graph. Setting $M = 10$ remains the best option for the vast majority of the trials across tested algorithms, diagnosis models and system sizes.

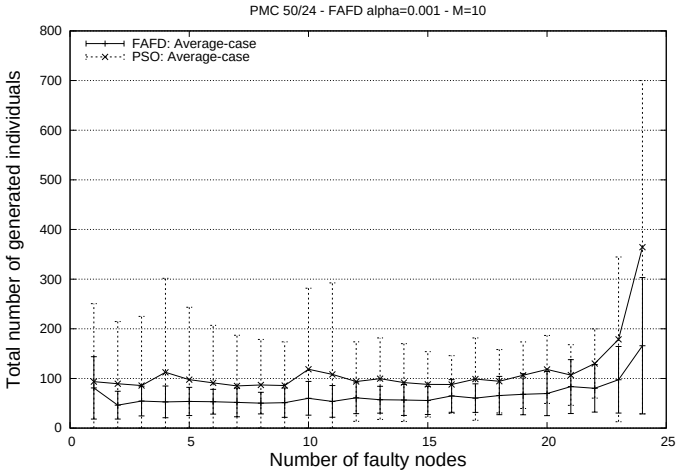
6.6.2 Experiment 1: PMC Model, Small Graph

Fig. 6.5(a) displays the ϕ indicator in presence of the invalidation model and a small system $G_{24}(50)$. The two steep AIS curves are due to the heavy effect of cloning, a vital mechanism to guarantee diversification in AIS along with the ensuing mutation operator. Because at every AIS iteration the highest ranked antibodies are cloned as many times as the population size itself [146], the associated memory requirements may be prohibitive for a device with limited resources. BPSO-FD and FAFD both exhibit a steady fault detection average trend as the number of damaged units grows, yet the latter requires fewer individuals than the former. FAFD's robustness is verified by the shorter confidence intervals depicted in Fig. 6.5(b). Moreover, the worst-case demeanor of BPSO-FD is not as encouraging as that of FAFD, with drastic rises in ϕ for $t \in \{4, 11, 23, 24\}$, as evidenced in Fig. 6.5(a). This is likely caused by a higher sensitivity to the random initial swarm, which is smoothed in FAFD via stronger interactions among the fireflies in their pursuit of the actual fault set.

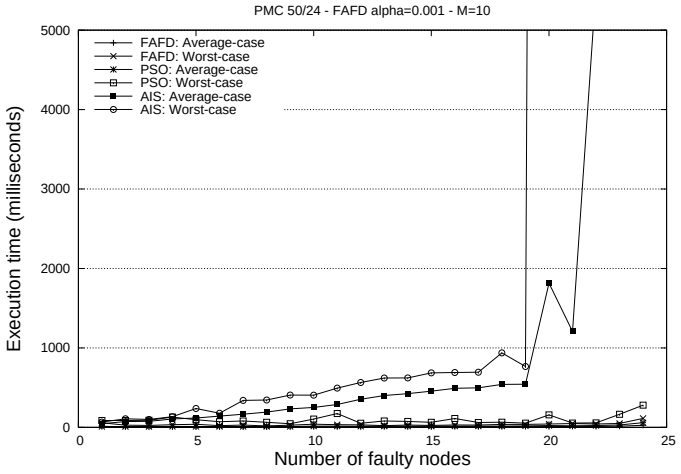
Concerning ρ, Fig. 6.5(c) helps us realize that only AIS fails to identify $\overline{F}$ in less than 5 sec. The large number of individuals spawned in AIS has an immediate impact on its com-



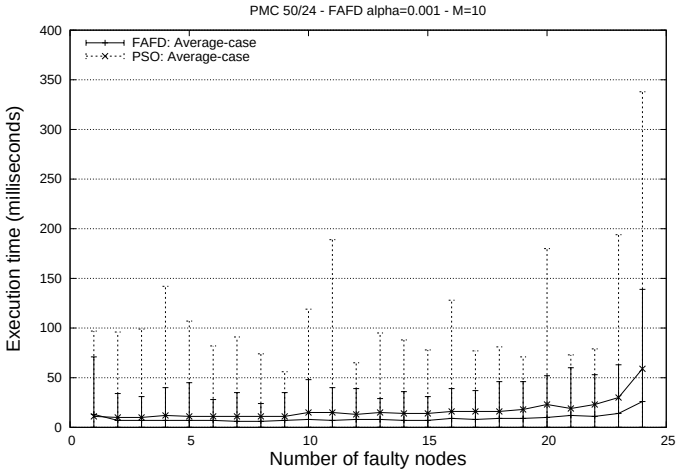
(a) ϕ vs. t



(b) FAFD vs. BPSO-FD: 95% confidence intervals of their average-case exploratory behavior



(c) ρ vs. t



(d) FAFD vs. BPSO-FD: 95% confidence intervals of their average-case convergence speed behavior

Figure 6.5: Performance of AIS, BPSO-FD and FAFD with the PMC model and $G_{24}(50)$ graph.

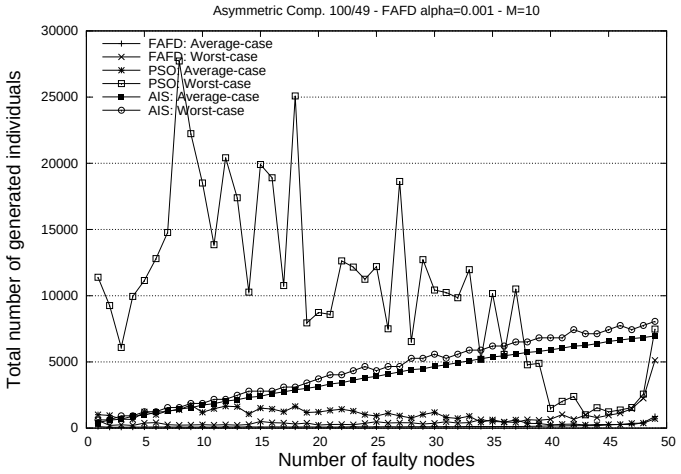
putational complexity, as represented by the two curves with the sharpest slopes. Conversely, the response time of the two swarm intelligence approaches remains nearly unaltered as more faulty nodes are added, which means that having at least a good particle direct the search of the remaining ones is profitable in small networks. Marginal improvements in favor of FAFD are disclosed when we look at Fig. 6.5(d). Again, there is more uncertainty in BPSO-FD's expected running times, as evidenced by the larger confidence intervals.

Similar results are obtained for the $G_{24}(50)$ in the symmetric comparison model, which are therefore left out of the discussion.

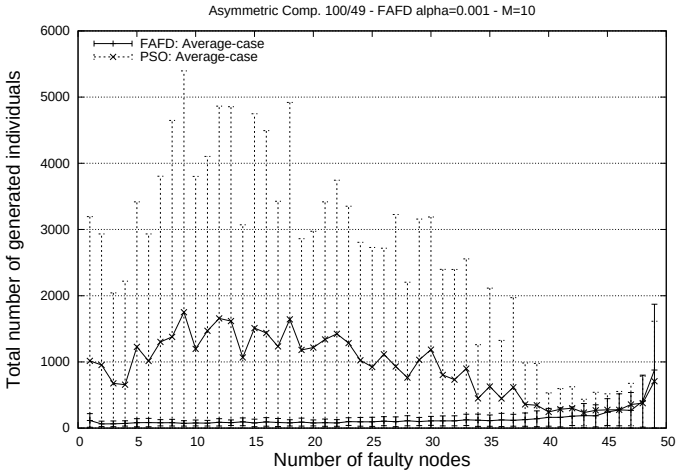
6.6.3 Experiment 2: Symmetric Comparison Model, Medium Graph

When the three algorithms are tested with a $G_{49}(100)$ graph, Fig. 6.6(a) brings up the issue of the irregular worst-case detection behavior suffered by BPSO-FD, as discussed before in Sect. 6.4.2 and 6.5. Fortunately, the joint influence of several informers together with the customized light absorption coefficient in (6.5) prevents FAFD agents from getting caught in local optima, as proved by the fact that FAFD consumes a nearly constant and negligible number of candidate solutions to converge. The evolutionary search undertaken by AIS yields almost identical results for the average and worst cases. Fig. 6.6(b) confirms that FAFD is preferred to BPSO-FD when it comes to memory consumption.

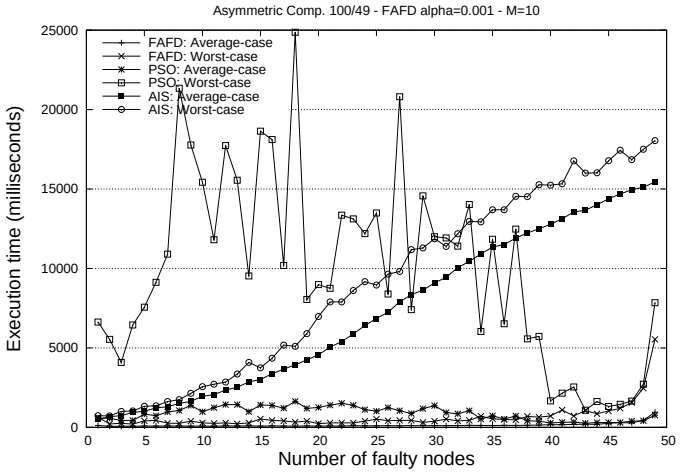
The running time of the firefly-based algorithm is smaller than that of rival methods. Fig. 6.6(c) displays how the unsatisfactory guidance in global-best BPSO-FD still produces a significant delay in convergence speed. Although BPSO-FD and FAFD do very well in the average case, for the former it takes up to 5x longer to detect $\overline{F}$ than the latter in the worst case. Nevertheless, most of the times both approaches efficiently reach the global optimum in less than 2 sec., as portrayed in Fig. 6.6(d). The reliability of the average time prediction for FAFD is manifested in the short length of most of its confidence intervals, with ample improvement over BPSO-FD.



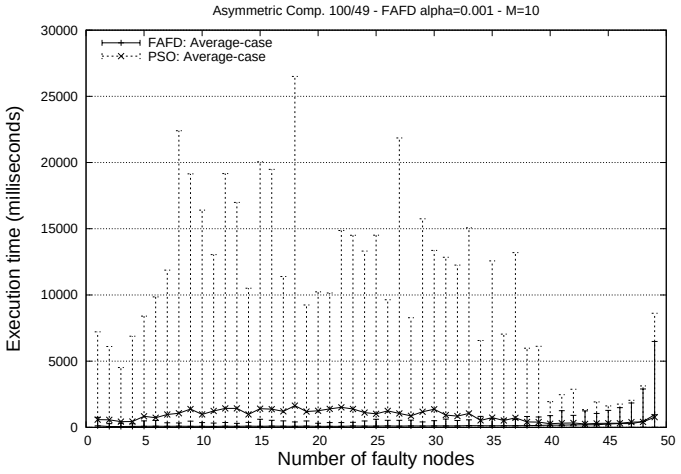
(a) ϕ vs. t



(b) FAFD vs. BPSO-FD: 95% confidence intervals of their average-case exploratory behavior



(c) ρ vs. t



(d) FAFD vs. BPSO-FD: 95% confidence intervals of their average-case convergence speed behavior

Figure 6.6: Performance of AIS, BPSO-FD and FAFD with the symmetric comparison model and $G_{49}(100)$ graph.

6.6.4 Experiment 3: PMC Model, Large Graph

Let us finally contrast FAFD and BPSO-FD in presence of a complex system $G_{249}(500)$. AIS was excluded from this experiment due to its poor and impractical behavior.

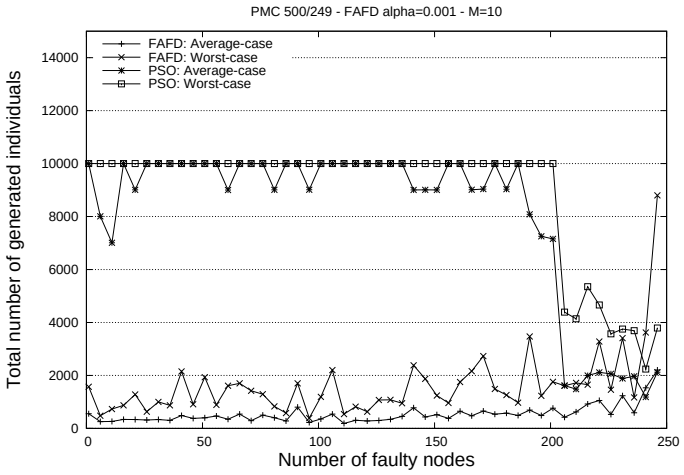
Fig. 6.7(a) bears witness to the inability of BPSO-FD in locating the true fault set for most $t \in \{1..200\}$, as depicted by the straight line topping ϕ along 1,000 iterations. Its vanishing effect can be explained by the same argument in Sec. 6.5. While FAFD never failed to encounter $\overline{F}$, BPSO-FD fell short in 78% of the cases. Recall that both methods are tried with a small population size ($M = 10$), which makes FAFD a strong candidate for facing challenging fault detection scenarios with low resource availability. Even FAFD's worst-case setting is preferable to BPSO-FD's average case. The dependability of FAFD is verified via Fig. 6.7(b) and Fig. 6.7(d).

BPSO-FD's running time ρ increases nearly monotonically in Fig. 6.7(c) with the expected number of faulty units until a large enough search space is reached for $t > 200$. In such cases, the existence of numerous feasible candidate sets helps attenuate the undesirable effect of having a single informer particle draw all remaining population members towards a reduced group of infeasible encodings. Because of BPSO-FD's high failure rate, its average and worst-case performances overlap to a great extent. Observe how FAFD responds with the true fault set within 20 sec. on average and always within 40 sec (except for $t = 249$). This fact can be ascribed to the enhanced social communication mechanisms present in the firefly swarm.

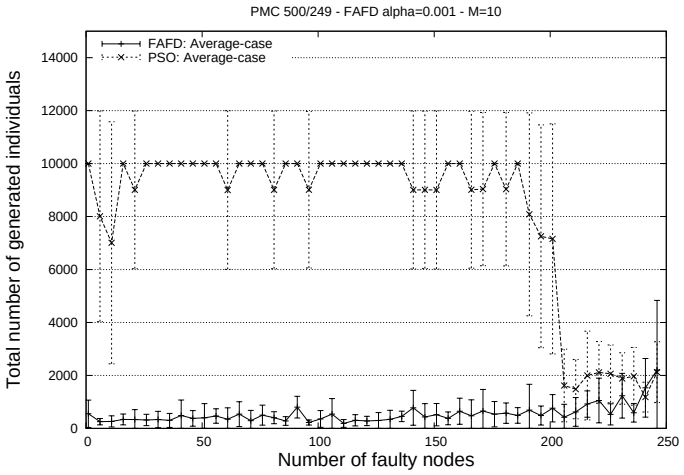
6.7 Conclusions

In this chapter we have demonstrated how fault identification in diagnosable systems with arbitrary topologies can be successfully carried out by swarm intelligence techniques once the problem is duly cast into the realm of combinatorial optimization.

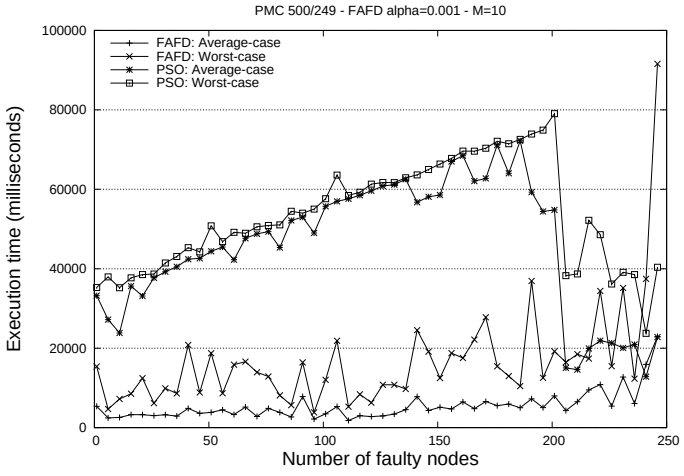
Artificial particles and fireflies with a binary encoding of the candidate fault sets flew all over the discrete search space guided by one (PSO) or multiple (FA) informers. Infeasible



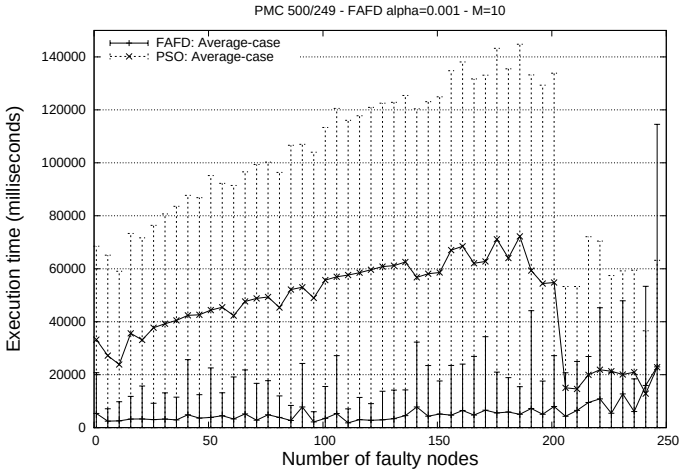
(a) ϕ vs. t



(b) FAFD vs. BPSO-FD: 95% confidence intervals of their average-case exploratory behavior



(c) ρ vs. t



(d) FAFD vs. BPSO-FD: 95% confidence intervals of their average-case convergence speed behavior

Figure 6.7: Performance of BPSO-FD and FAFD with the PMC model and $G_{249}(500)$ graph.

solutions were repaired by flipping the assumption on the status of the least promising nodes.

To overcome the deficiencies exhibited by BPSO-FD in the worst-case prediction scenario in medium and large networks, a stronger degree of social interaction among the search agents was required. This breakthrough was achieved through the Firefly Algorithm once its swarm members were forced to navigate in a personalized search subspace, whose characteristic length was embedded into an adaptive light absorption coefficient. As a result of such design guidelines, we witness an accelerated convergence to the global optimum and a very low memory consumption rate (in contrast to BPSO-FD and AIS) under the invalidation and comparison test models and regardless of the system size.

The next chapter introduces our second framework for identification of the faulty nodes.

CHAPTER 7

AN EVOLVING RISK MANAGEMENT FRAMEWORK

In the previous chapter, the distributed framework of *diagnosable systems* was adopted for identifying the set of faulty units in the WSN. Two swarm intelligence metaheuristics that lower the memory requirements during the search process and converge fast to the true fault set were unveiled as part of the thesis contributions.

In spite of the encouraging results attained with system-level fault diagnosis, the previous framework possesses some intrinsic limitations that make it a rather rigid and machine-driven alternative. These drawbacks were briefly mentioned in Sect. 1.2.3.

First, the fault identification process in diagnosable systems of arbitrary topology is permeated with an uncertainty issue that must be carefully handled. This is owing to the fact that the sensing units in such a framework are responsible for conducting the testing phase altogether. Because both faulty and fault-free nodes are allowed to diagnose the status of their peers, the unreliable nature of the tests performed by the faulty units makes it imperative to submit all test outcomes system-wide to a supervisory (fault-free) entity, which derives the true status of each sensor and assigns a label (faulty/ fault-free) to it. This supervisory entity could be the base station (sink) or a gateway (cluster-head) node, given that the two algorithms in Chapter 6 that probe the search space of all faulty sets need a minimal amount of resources (memory, computing power and running time) to do so.

Second, system-level fault diagnosis forces the risk assessment of any sensor to take a binary form (faulty or fault-free), thus leaving enough ground for concerns on the extent of the risk posed by the sensing unit at any point in time. A more descriptive evaluation of the goodness of the sensing unit (e.g. through a quantification of its overall risk) could prove more helpful to the decision-making process.

Third, the distributed model in Chapter 6 bears a rigid and non-adaptive structure. Fault identification occurs in a highly automated fashion once the manifold risk factors that a sensing unit is exposed to (e.g. battery level, proximity to a hazardous source, inconsistency of the measurements over time, etc.) have been hardwired a priori into the sensor's architecture. A time-evolvable risk perception is hard to assimilate by a diagnosable system and the communication overhead entailed by broadcasting the changes in the local risk perception to all nodes in the system is non-negligible.

The fault detection framework studied in Chapter 6 could be complemented with a more human-centric counterpart. While automation in the fault identification task continues to be necessary so that the WSRN can still work unattendedly, we would like the system to gladly override its understanding of risk (either for a particular sensor or globally) with that provided by a human expert who is occasionally monitoring it.

There is little work in the literature when it comes to *risk management frameworks for WSNs* and our expectation is to narrow that gap. This chapter introduces a novel risk management framework with a multi-modular architecture that evolves the local risk perception over time. Though its overall design is largely automated, it enables a human expert who occasionally monitors the system to override the current formulation of risk, either globally or for a particular sensor. This is possible through the interaction with the *risk visualization* and *risk assessment* modules, which are centered around the shadowed set formalism [117].

The visualization module displays the overall risk per system unit in the form of clusters corresponding to the sensor nodes in the risk feature space. The clustering results vary over time in order to reflect the WSN dynamics. Shadowed sets are a natural extension to fuzzy

sets and we used them to represent evolving clusters given their robustness against noise and outliers as well as their fast convergence and high interpretability [105]. In the risk assessment module, fuzzy and shadowed evaluations of the total risk of any system unit are provided. The framework under consideration was successfully applied to a simulated Critical Infrastructure Protection scenario and becomes also an integral part of the coverage formation and repair algorithms discussed in Chapters 3 - 5.

Compared to diagnosable systems, the risk management framework unrolled in this chapter is more flexible, human-centric and adaptive, although it leans more heavily on a centralized analysis of the network dynamics by the WSN manager.

Sect. 7.1 offers an overview of the risk-driven platform. The risk visualization and assessment modules are dissected in Sect. 7.2 and 7.3, respectively. Experiments are unfolded in Sect. 7.4 and conclusions stated in Sect. 7.5.

7.1 Risk Management Framework for WSNs

Fig. 7.1 displays the architectural view of the proposed risk management framework.

Although continuous data will arrive in real-time to the centralized location (e.g. base station) where our risk-driven framework is running, the stream is split into multiple discrete snapshots in order to better capture the structural dynamics of the WSN at any time slot. That is to say, a *data snapshot* is the main structural piece of information that all the risk-aware modules in the framework will deal with. The size of a snapshot is highly problem-specific. It could hold, for example, risk-related parameters for each sensing unit in the WSN.

The first step is an initial characterization of the set of risk features $F = \{F_1, F_2, \ldots, F_n\}$ that the system will monitor. This is accomplished through the *risk feature extraction* module that operates with live, raw sensor data coming from the WSN deployment field. Sensor readings may immediately become risk features (e.g. battery level) or they may need to undergo some transformation (e.g. sonar and laser range finder measurements can be amalgamated to

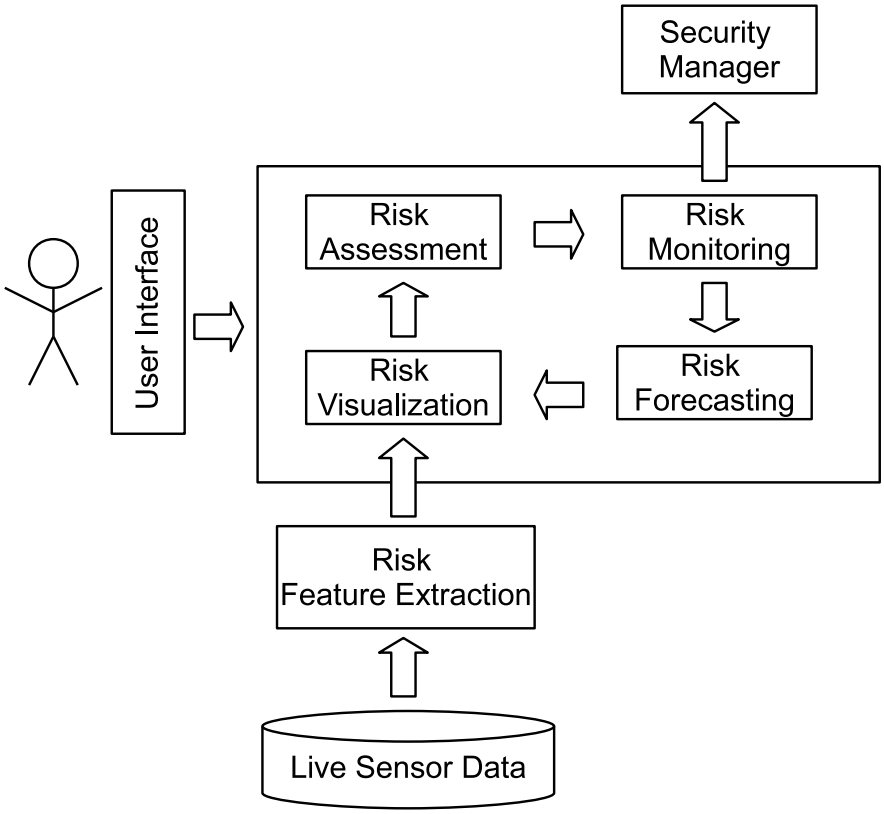


Figure 7.1: The architecture of the evolving risk management framework.

provide a more accurate estimate of the distance from a potential intruder).

Risk visualization allows the user to watch clusters corresponding to the sensor nodes in the risk feature space. By doing so, the human expert (e.g. WSN manager) is able to alter the level of the perceived threat for a particular sensor or group of sensors. This knowledge is immediately incorporated into the system. Cluster visualization is a general-enough representation of the system's prevalent dynamics and thus can be applied to any risk source (e.g. cyber-attack, measurement error, etc.)

Risk assessment quantifies the overall risk posed by any system unit on the basis of the local risk across all indicators. It is sensitive to adaptive user feedback in modeling the local risk perceptions and adjusts its behavior upon any observable sensor failure.

Risk monitoring probes the output of the previous module and triggers alarms whenever a threatening condition arises. The simplest strategy is a thresholding technique, in which an alarm goes off for any sensor whose current risk level exceeds a tolerable cap. More sophisticated alarm generation techniques could be enforced in this module as well.

Risk forecasting updates its built-in model for risk prediction on the basis of the observations in the current snapshot. It allows to predict future risk trends and hence becomes a crucial input for the decision-making process.

Security manager coordinates actions that must be taken upon the environment to mitigate the danger, e.g. replacing a damaged sensor or sprinkling water over a blazing area.

This chapter elaborates solely on the risk visualization and assessment modules. They fit well in the general scope of the doctoral thesis (i.e. the development of fault-reactive WSNs) and are enough to provide the basic services needed to fulfill the thesis goals enunciated in Sect. 1.2. The remaining modules will be researched as part of future post-doctoral endeavours in the context of a risk-driven decision support tool funded by industrial partners.

7.2 The Risk Visualization Module

Evolving fuzzy clustering [118] has been recently spotted as a promising tool to learn knowledge structures (clusters) in dynamic environments. Our visualization module hinges on an evolving clustering approach based on the offline Shadowed C-Means (SCM, see Sect. 2.1.3) method [105]. The structure and number of the clusters may change across any two consecutive snapshots in order to better reflect the reality of the WSN at any moment in time.

7.2.1 Evolving Shadowed Clustering (ESC)

In the following, we outline the algorithmic steps of the evolving clustering methodology rooted on shadowed sets.

Initialization

When the first data snapshot $ii = 1$ arrives, an initial number of clusters $c[1]$ has to be picked from the user-specified interval $[c_{min}, c_{max}]$. SCM is run multiple times per value of $c[1]$, and ESC retains the one that maximizes on average the *quality of the clustering*, as shown in (7.1)

$$\gamma[ii] = \frac{\sum_{j=1}^{c[ii]} | \, CORE(V_j[ii]) \, |}{1 + | \, \bigcup_{j=1}^{c[ii]} SDW(V_j[ii]) \, |} \tag{7.1}$$

where $V_j[ii]$ is the prototype of the j-th cluster at the ii-th data snapshot[1]. This $\gamma[ii]$ indicator rewards clusterings with a high number of patterns in the core regions and a small number of patterns in the shadowed regions. In other words, it aims at minimizing the amount of uncertainty on the membership of patterns to clusters in the present snapshot.

Upon arrival of any subsequent snapshot, we run SCM with exactly $c[ii-1]$ clusters to uncover the current structure. Further adjustments might be needed, as explained below.

[1] If the snapshot number is omitted, it means the current snapshot ii.

Cluster Metrics

For each cluster, we compute its accuracy α, compactness ρ and stability θ metrics as follows:

$$\alpha(V_j) = \begin{cases} -\infty & |\,\mathrm{CORE}(V_j)\,| + |\,\mathrm{SDW}(V_j)\,| = 0 \\ \dfrac{|\,\mathrm{CORE}(V_j)\,|}{|\,\mathrm{CORE}(V_j)\,| + |\,\mathrm{SDW}(V_j)\,|} & \text{otherwise} \end{cases} \tag{7.2}$$

$$\rho(V_j) = \begin{cases} 1 & \mathrm{SDW}(V_j) = \emptyset \\ 0 & \mathrm{CORE}(V_j) = \emptyset \\ \dfrac{\dfrac{1}{|\,\mathrm{SDW}(V_j)\,|} \displaystyle\sum_{X_k \in \mathrm{SDW}_{(V_j)}} u_{jk}}{\dfrac{1}{|\,\mathrm{CORE}(V_j)\,|} \displaystyle\sum_{X_k \in \mathrm{CORE}_{(V_j)}} u_{jk}} & \text{otherwise} \end{cases} \tag{7.3}$$

$$\theta(V_j) = w \cdot \alpha(V_j) + (1 - w) \cdot \rho(V_j) \tag{7.4}$$

The *accuracy* α of a cluster V_j indicates the proportion of representative patterns in it. Accuracy is defined as the ratio of the number of patterns in the core region to the number of patterns in the core and shadowed regions. The closer this indicator to 1, the stronger the cluster definition is.

The *compactness* ρ denotes how far in membership the core and shadowed zones of a cluster are on average. As this measure approaches 1, it means that the gap in the average membership of those data patterns in the core and shadowed regions is narrowing down, which indirectly speaks of a geographical proximity of the data patterns in the risk feature space. The λ threshold associated with that cluster is what virtually separates the patterns into the core and shadowed zones yet they are members of a fairly compact knowledge structure.

The *stability* θ amalgamates both measures according to the user-defined importance of the cluster accuracy, which is denoted by the w parameter.

Dynamic Cluster Formation

This stage performs outlier detection as well as cluster deletion, split and merge. A reclustering via SCM is triggered should the current cluster distribution change due to any of the following operations:

Outlier detection: A data pattern X^* is flagged as an outlier (and thus becomes a new cluster prototype) if and only if:

(1) It does not belong to the core region of any cluster.

(2) It lies farthest from every cluster prototype than any pattern in that cluster.

(3) The distance to the nearest pattern in its cluster exceeds the maximal intra-cluster distance (excluding itself).

The outlier detection mechanism enforced in this implementation combines ideas from distance-based and density-based methods. An example is depicted in Fig. 7.2.

Cluster deletion: Let ϵ^- and N_1 be two built-in parameters named "stability threshold" and "minimal granularity". For each cluster V_j with $\theta(V_j) < \epsilon^-$, delete V_j if $\text{CORE}(V_j) = \emptyset \wedge |\text{SDW}(V_j)| < N_1$. This step ensures that out-of-date and noisy structures at the current snapshot are dropped by the system.

Cluster split: For each cluster V_j with $\theta(V_j) < \epsilon^-$ and not deleted in the previous step, split it. That is, compute two new cluster prototypes V_s and V_t so that $s = \arg\max_{X_k} u_{jk}$ and $t = \arg\min_{X_k} u_{jk}$. In other words, the most and least representative data patterns of V_j become the two new cluster prototypes. A cluster split is indicative of a *drift* in its structure that may have begun several data snapshots ago.

Cluster merge: Two clusters with prototypes V_s and V_t are merged if their degree of overlap exceeds the ϵ^+ threshold. This process is repeated until there are no more clusters to merge. The overlap degree $\text{O}(V_s,V_t)$ is calculated as in (7.5).

$$O(V_s, V_t) = 1 - \frac{A_1 + B_1 + C_1 + D_1}{|\,\text{CORE}(V_s)\,| + |\,\text{CORE}(V_t)\,| + E_1 + F_1} \tag{7.5}$$

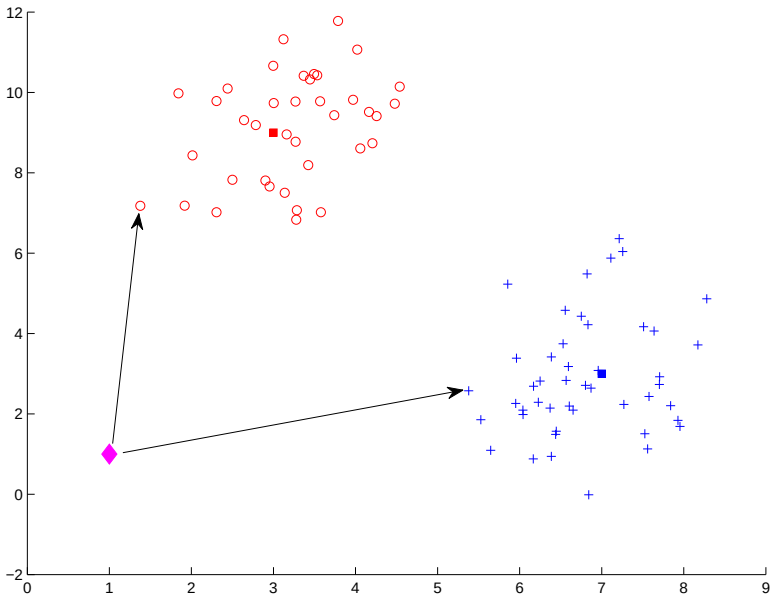


Figure 7.2: Illustration of the outlier detection mechanism. The data pattern represented by the purple diamond is labeled as an outlier and becomes a new cluster prototype. A solid arrow indicates the distance to the nearest member in a cluster whereas a filled square symbolizes a cluster prototype.

where

$$A_1 = \sum_{X_k \in \text{CORE}(V_s)} (1 - u_{tk}) \qquad B_1 = \sum_{X_k \in \text{CORE}(V_t)} (1 - u_{sk})$$

$$C_1 = \sum_{X_k \in \text{SDW}(V_s) \cap V_s} (u_{sk} - u_{tk}) \qquad D_1 = \sum_{X_k \in \text{SDW}(V_t) \cap V_t} (u_{tk} - u_{sk})$$

$$E_1 = \sum_{X_k \in \text{SDW}(V_s) \cap V_s} u_{sk} \qquad F_1 = \sum_{X_k \in \text{SDW}(V_t) \cap V_t} u_{tk}$$

Notice that only the representative patterns in both clusters are considered in the calculation of their overlap. During merge, V_s and V_t amalgamate into a single prototype V_j which takes the form in (7.6):

$$V_j = \frac{A_2 + B_2 + C_2 + D_2 + E_2}{\mid \text{CORE}(V_s) \cup \text{CORE}(V_t) \mid + F_2 + G_2 + H_2 + I_2} \tag{7.6}$$

where

$$A_2 = \sum_{X_k \in \text{CORE}(V_s) \cup \text{CORE}(V_t)} X_k \qquad B_2 = \sum_{X_k \in \text{SDW}(V_s)} u_{sk}^m X_k$$

$$C_2 = \sum_{X_k \in \text{SDW}(V_t)} u_{tk}^m X_k \qquad D_2 = \sum_{X_k \in \text{XCL}(V_s)} u_{sk}^{m^m} X_k$$

$$E_2 = \sum_{X_k \in \text{XCL}(V_t)} u_{tk}^{m^m} X_k \qquad F_2 = \sum_{X_k \in \text{SDW}(V_s)} u_{sk}^m$$

$$G_2 = \sum_{X_k \in \text{SDW}(V_t)} u_{tk}^m \qquad H_2 = \sum_{X_k \in \text{XCL}(V_s)} u_{sk}^{m^m}$$

$$I_2 = \sum_{X_k \in \text{XCL}(V_t)} u_{tk}^{m^m}$$

Shift Detection

A *shift* is a rather abrupt transition of a cluster in the feature space across two consecutive snapshots. The pattern composition of the shifted cluster before and after the event is very similar. To be aware of shift occurrences, ESC performs a cluster matching between $V_s[ii-1]$ and $V_t[ii]$ if and only if $|\,\mathrm{CORE}(V_s[ii-1]) \cap \mathrm{CORE}(V_t[ii])\,| > |\,\mathrm{CORE}(V_s[ii-1])\,|/2$, i.e. cluster $V_t[ii]$ holds at least half of the most significant patterns of $V_s[ii-1]$. Let P be the total number of matched pairs. Then the average translation distance d_{avg} is defined as:

$$d_{avg} = \frac{1}{P} \sum_{(V_s[ii-1], V_t[ii]))} \mathrm{dist}(V_s[ii-1], V_t[ii]) \tag{7.7}$$

where $\mathrm{dist}(\cdot)$ is any distance metric. ESC triggers a shift alarm for every matched pair where $\mathrm{dist}(V_s[ii-1], V_t[ii]) > \max(d_{avg}, 0.1)$.

Cluster Variation Metrics

For each matched pair $(V_s[ii-1], V_t[ii])$, the variation metrics below are calculated:

$$\begin{aligned}
\Delta\alpha(V_s[ii-1], V_t[ii]) &= \alpha(V_t[ii]) - \alpha(V_s[ii-1]) \\
\Delta\rho(V_s[ii-1], V_t[ii]) &= \rho(V_t[ii]) - \rho(V_s[ii-1]) \\
\Delta\theta(V_s[ii-1], V_t[ii]) &= \theta(V_t[ii]) - \theta(V_s[ii-1])
\end{aligned}$$

Clustering Metrics

As to the resultant clustering, we would like to assess its quality and the extent of the structural variations occurring from the previous snapshot. The former is achieved with the γ measure in (7.1) and the latter as calculated in (7.8).

$$\Psi[ii] = \sum_{s=1}^{c[ii-1]} \sum_{t=1}^{c[ii]} \psi(w_{st}) \tag{7.8}$$

where w_{st} is the membership grade of $V_t[ii]$ to $V_s[ii-1]$ computed as in (2.11) whereas

$\psi(\cdot)$ is a functional that quantifies the degree of structural variation of $V_t[ii]$ with respect to the shadowed set representing $V_s[ii-1]$. Fig. 7.3 shows its schematic portrayal. There is no structural variation if the prototype in the present snapshot falls within the core or the exclusion regions of the shadowed set associated with the cluster prototype in the previous data snapshot. The variation degree increases as we approach 0.5, the focal point of the shadowed region.

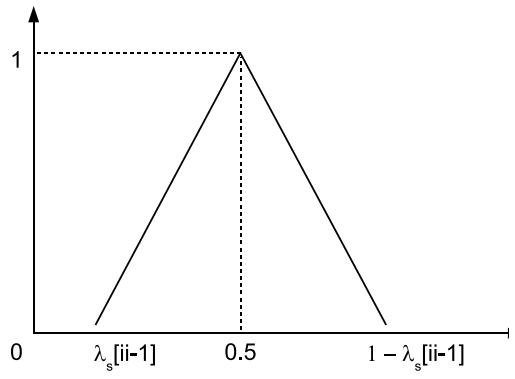


Figure 7.3: Membership function of $\psi(w_{st})$. The threshold $\lambda_s[ii-1]$ is used.

Handling the Single-Cluster Case

During the dynamic cluster formation process above, it could be that the deletion and merge operations drive the entire cluster distribution for a given snapshot towards a single final structure V. This is an interesting formation in the sense that no new clusters can be spawned out of it, for $\mu_V(X) = 1\ \forall X \in U$ according to the classical fuzzy clustering machinery in (2.11).

To deal with this special case, we lean on the simple concept of *cluster dispersion* quantified via the distances from data patterns to prototype V. Let $\overline{\text{disp}}$, disp^+ and disp^- be the average dispersions of the entire cluster, its N_1 farthest patterns and its N_1 nearest ones, respectively. Cluster X is split if and only if $\text{disp}^+ - \text{disp}^- > \overline{\text{disp}}$ and its farthest element X^+ is chosen as the prototype of the newborn cluster.

7.3 The Risk Assessment Module

Given an input vector $\vec{X} = (x_1, x_2, ..., x_n)$ with the measurements of a particular sensor S in the risk feature space, a straightforward manner to assess the overall risk of S at any point in time is by using a rule-based system [95]. Instead of formulating crisp rules, we would like to have some sort of information granule [116] in the rules' antecedents and consequents in order to make the risk assessment module human-friendly and interpretable.

In the context of a risk-driven evaluation of any sensing unit in a WSN, the rule base could look like this:

$$\begin{aligned} &\text{IF RISK}(F_1) \text{ is HIGH OR } \ldots \text{ RISK}(F_n) \text{ is HIGH} \\ &\qquad\qquad \text{THEN TOTAL\_RISK}(S) \text{ is HIGH} \end{aligned} \tag{7.9}$$

where $\text{RISK}(F_i)$, $i = 1..n$ is an information granule (either a fuzzy or a shadowed set) whose membership function $\mu_{X_{F_i}}(\cdot)$ (and λ_i threshold, if needed) is set by the user according to general expert domain knowledge. Should the risk perception change over time (which is generally not common in WSNs), these granules are updated by the user accordingly. For the rule base illustrated in (7.9), the valuation of the overall risk τ in the rule's consequent obeys the expression (7.10).

$$\tau = \max_{i=1}^{n} \{w_i \cdot \varphi(\mu_{X_{F_i}}(x_i))\} \tag{7.10}$$

where w_i is the weight of F_i and $\varphi(\cdot)$ is a mapping from the fuzzy membership grade space to the fuzzy or shadowed membership grade space as shown in (7.11).

$$\mu_{X_{F_i}}(x) = \begin{cases} x & \text{if } X_{F_i} \text{ is a fuzzy set} \\ 1 & \text{if } X_{F_i} \text{ is a shadowed set AND } x \in \text{CORE}(X_{F_i}) \\ 0 & \text{if } X_{F_i} \text{ is a shadowed set AND } x \in \text{XCL}(X_{F_i}) \\ x & \text{otherwise} \end{cases} \tag{7.11}$$

The shadowed-set-based estimate behaves proactively if any local risk condition is threatening enough and dismisses locally irrelevant risk reports so that the system does not overreact to small deviations in the reported mesurements. Feature weights are user-provided at the outset and adapted afterwards by means of a simple learning scheme. That is, upon an observable sensor failure caused by risk pattern $\vec{X}$, the weights of the risk features are raised proportionally to their perceived danger to the sensor, as displayed in (7.12).

$$w_i = w_i + \frac{\mu_{X_{F_i}}(x_i)}{\sum_{i=1}^{n} \mu_{X_{F_i}}(x_i)} \cdot w_i \tag{7.12}$$

Notice that the number and structure of the rules (i.e. connectors for the antecedents, aggregation for the consequent, etc.) is highly problem-dependent. In the Territorial Security outdoor scenario unveiled in Sect. 7.4, our singleton-rule-based assessment module was able to finely capture the potential threats of the application though.

7.4 Experimental Study

The evolving risk management framework under discussion has been applied to a novel cooperative networking scenario depicted in [52], in which a mobile robot augments the wireless sensor network with self-healing capabilities by repairing its coverage whenever a damaged sensor stems. The simulations have been conducted in Microsoft Robotics Development Studio (MRDS)[2] on an Intel Core i7 CPU 860 @ 2.80 GHz with 6 GB of RAM under Windows 7 Home Premium with a 64-bit architecture.

A territorial perimeter around a critical infrastructure is set up by laying out a group of stationary platforms (in white), as shown in Fig. 7.4. The brown node outside the perimeter is an intruder which will try to compromise the secured area. Each platform is equipped with a built-in, battery-powered sonar device, which allows monitoring its immediate vicinity. The ID of a platform equals that of its only-mounted sensor. Two risk sources (features) per

[2] http://www.microsoft.com/robotics/

deployed platform are considered: the *battery level* and the *intrusion distance*, each bearing an associated risk perception. A data snapshot containing the feature vector reported by each platform in the outdoor environment is processed centrally every 60 ms.

Whenever the total risk for a sensing unit S goes beyond a permissible threshold (as portrayed in Fig. 7.5) the mobile robot located inside the perimeter (see Fig. 7.4) will add the coordinates of S to its visiting list and eventually move there to better cover the environment in an attempt to detect the intruder and take further actions. The robot also replenishes the battery of the visited sensor.

Figure 7.4: The simulated outdoor Territorial Security scenario.

The user-specified initial fuzzy modeling of the risk features is displayed in Fig. 7.6 and 7.7. While the risk linearly rises with the decline of battery power (Fig. 7.6), the trapezoidal membership function in Fig. 7.7 indicates that the presence of the intruder within a 2-m radius of any peripheral node has maximal bearing and it gradually vanishes until completely fading away after 3 m. The graphical interface allows the user to select an alternative shadowed modeling for any risk feature, in which case the associated λ threshold is to be properly set.

Figure 7.5: Feature-dependent and overall risk assessment for each sensor in the field.

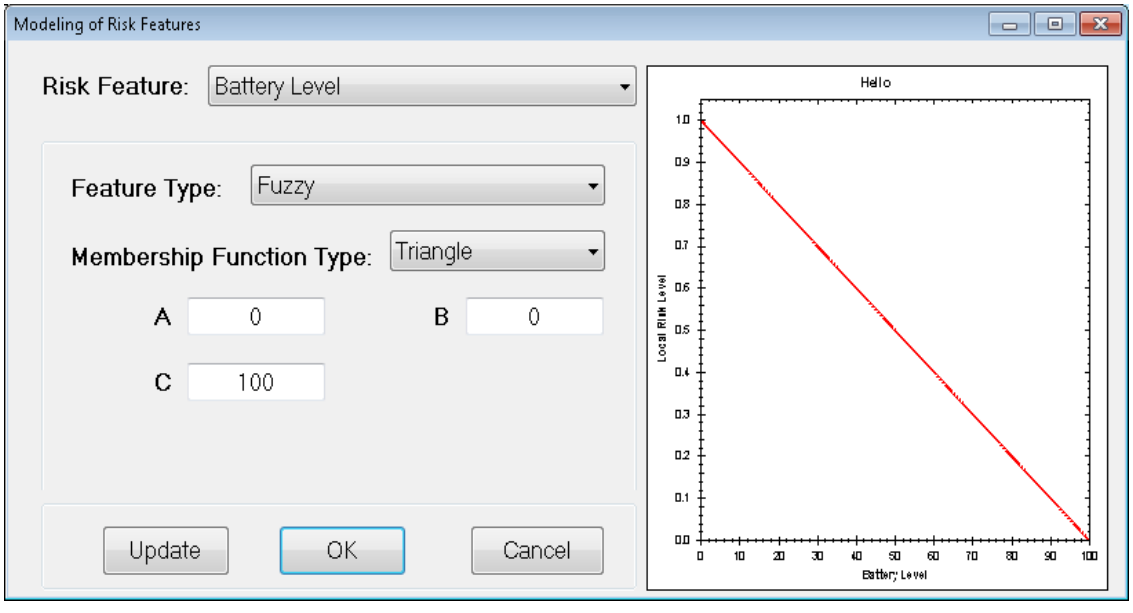


Figure 7.6: Fuzzy set specification of the battery level risk.

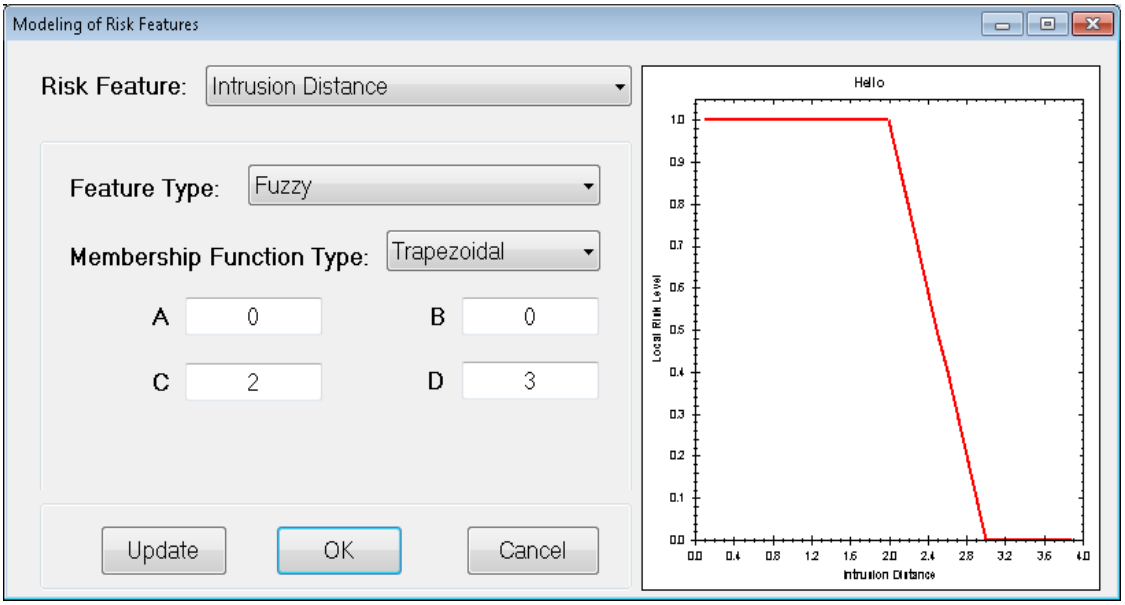


Figure 7.7: Fuzzy set specification of the intrusion distance risk.

In parallel with the individual risk assessment of each system unit, the risk visualization module governed by the ESC algorithm is offering a visual feedback to the user about the structural dynamics of the WSN. Concept drifts and shifts are discovered during ESC's execution by comparing the cluster distributions of the previous and current data snapshots guided by the metrics in Sect. 7.2.1, as displayed in Fig. 7.8.

The cluster-related metrics in Sect. 7.2.1 serve as faithful indicators of the robustness of the uncovered structures. Whenever an unexpected event takes place in the system (e.g. a transition into an abnormal functioning mode or the nearness of the intruder to the network perimeter), this is immediately reflected in the accuracy, compactness and stability of the affected clusters. The dynamic cluster formation process outlined in Sect. 7.2.1 will reaccommodate the cluster distribution by updating the cluster prototypes as well as creating or deleting clusters so that a stable set of knowledge structures can be attained in the current data snapshot. This is illustrated in Fig. 7.9.

Table 7.1 lists all ESC parameters. Their values have been determined after a thorough empirical analysis in which relevant sampling ranges for each of them were considered.

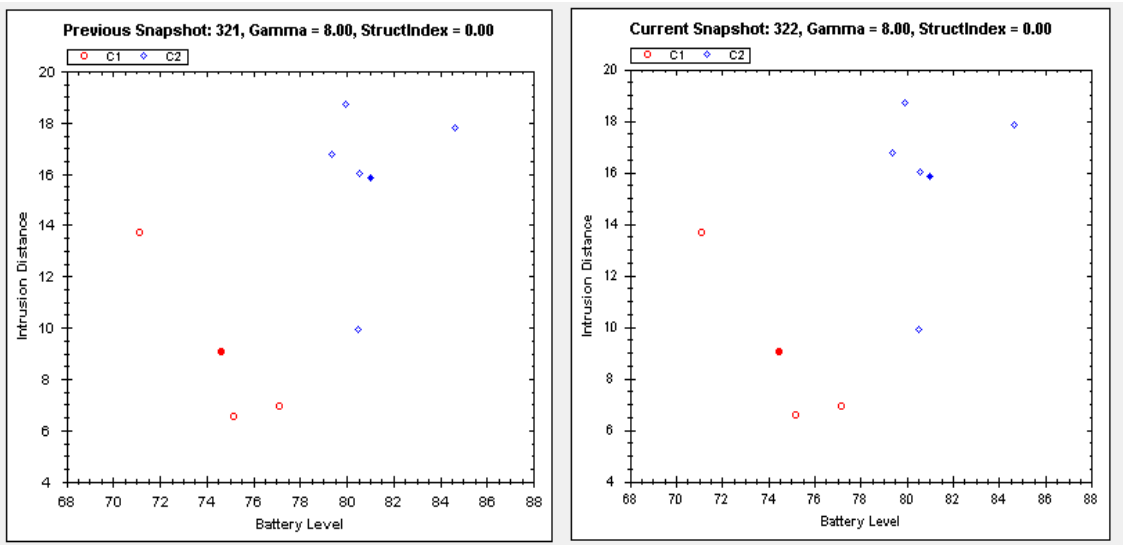


Figure 7.8: Two consecutive data snapshots being processed by the Evolving Shadowed Clustering (ESC) architecture.

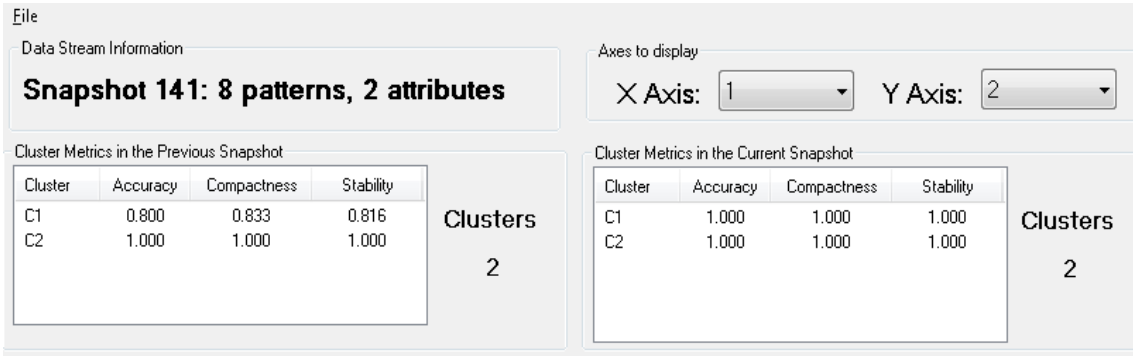


Figure 7.9: Cluster-related metrics computed by the ESC for each data snapshot. Notice how the loss of stability in the previous snapshot for C_1 has been overcome in the current snapshot by means of the dynamic formation of clusters, in this case through the update of the cluster prototypes.

Table 7.1: ESC Parameters

Parameter	Value	Description
m	2	Fuzzifier for SCM
T_{max}	100	Maximum number of SCM iterations
$[c_{min}; c_{max}]$	[2; 10]	Interval for the initial number of clusters
ϵ^{-}	0.4	Stability threshold
ϵ^{+}	0.35	Overlap threshold
w	0.5	Accuracy weight
N_1	3	Minimal number of patterns in a cluster

7.5 Conclusions

We have introduced a multi-modular, real-time risk management framework for WSNs featuring an evolving clustering architecture based on the shadowed set formalism and a fuzzy/shadowed risk assessment module with a simple learning procedure for weight update. The successful application of the proposed methodology to a Territorial Security scenario proved its feasibility in terms of bringing real-time risk awareness into the decision making process inherent to any parallel or distributed system. The framework under consideration is the first of its kind in the literature that provides a general-enough and visually appealing schematic representation of the WSN dynamics in the form of clusters of sensor nodes. It also stands as a more adaptive and human-centric alternative to the one based on diagnosable systems that was described in Chapter 6.

The next chapter wraps up the results accomplished across this doctoral investigation and voices future directions for extending the undertaken research efforts.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

This chapter summarizes the research endeavours conducted thus far and discusses some directions worth pursuing after the thesis.

8.1 Conclusions on the Present Work

The goal of this thesis, as described in Sect. 1.2, is to endow wireless sensor networks with a high degree of reactiveness when unexpected faults arise in the sensory nodes. This is accomplished by means of the locomotion and carrying abilities of robotic agents.

In Chapter 3, mobile robots receive real-time notifications (both during and after deployment of the tessellation layout around the POI) from functional units about the coordinates of defective nodes so that they can quickly move there and drop another sensor, thus preserving the coverage radius of the network. This process is carried out asynchronously at each node, with fine-grained inter-robot, inter-sensor and robot-sensor communication mechanisms playing a vital role in the energy-efficient nature of the CBFCF localized algorithm.

Responsiveness to sensor faults was also attained in a post-deployment scenario, where a team of mobile robots follow a nearly-optimal trajectory (computed at the base station through bio-inspired metaheuristic approaches) to relocate previously picked passive units all over the

field to the spots where the damaged nodes lie. The energy spent in failure reporting from each sensor node to the base station is compensated with a prolonged network lifetime made possible by the novel RA-WSN sensor relocation framework in Chapters 4 and 5. The three algorithms put forward are able to find good solutions in a very limited time span.

The reliable identification of the defective units is a crux in the picture of fault-reactive WSNs. For that reason, the two aforesaid scenarios are to be augmented with appropriate methodologies for the recognition of faulty units. The system-level fault diagnosis framework described in Chapter 6 and the risk management platform formalized in Chapter 7 are indeed two suitable and efficient avenues to highlight the need for replacement of any sensing unit. Any of these approaches can be run at a supervisory entity. For example, a cluster-based topology could be easily configured for the WSN in Chapters 3 - 5, where nodes in the same cluster test each other and report the outcomes to their cluster head, which in turn runs any efficient fault identification algorithm of the two presented in Chapter 6. Alternatively, the cluster members may send their individual risk-related information to their corresponding cluster head, which then performs the risk feature extraction upon the received data and assesses the level of risk posed by that particular sensor. Once the damaged (riskiest) nodes are identified, the cluster head reports their coordinates via multi-hop routes to the robotic actuators (in the focused coverage case) or to the base station (in the coverage repair case).

The ensemble of contributions depicted in this doctoral study is just a step forward in the feasible realization of enduring, trustworthy surveillance scenarios thanks to the seamless integration of robotic and sensory nodes. Our aim is to bring awareness to the scientific community of this promising, still-infant area of fault-reactive WSNs and motivate a wide assortment of further research works on this topic.

8.2 Further Work

In alignment with the overall thesis goal, we feel that the following research directions should be undertaken in the near future so as to provide a truly coherent portrait of a fault-reactive wireless sensor network.

8.2.1 On Robot-Dependent Focused Coverage Formation

The robot-dependent focused coverage formation problem around a POI presented in Chapter 3 could be extended to a more complex environment where a series of continuous POIs exists, thus forming a Line of Interest (LOI). This could represent, for example, the trace of a certain event or the boundary of a landmark (e.g. a building). Multiple discrete target objects, each modeled after a POI or a LOI, add more intricacy to the problem. The research goal would then be to derive an approach that preserves CBFCF's desirable features and attains some sort of shared coverage between nearby targets, possibly having irregular shapes, hence reducing the network size and eliminating harmful wireless interference. We know not of any protocol that targets such a pragmatic and entangled scenario.

8.2.2 On Robot-Assisted Coverage Repair

There is no indication thus far in Chapter 4 that the passive units collected all over the field by the mobile robot are to last long once deployed at the coordinates of the damaged sensors. The current problem formulation is only concerned with the optimization of the total distance traveled by the robotic agents. In other words, shorter sensor relocation trajectories are preferred. However, for the sake of prolonged network lifetime we may choose longer paths if the passive sensors therein involved have higher battery power. These two often-conflictive goals (path durability vs. path length) could be balanced through a multi-objective optimization approach, possibly leaning again upon the concurrent exploratory power of nature-inspired metaheuristics. Ongoing research efforts have been undertaken in this area already.

A decentralized implementation of the robot-team-assisted coverage repair scenario in Chapter 5 is also desirable, given that all robots do not necessarily need to lie at the base station but may be rather scattered throughout the field. From a localized viewpoint, we may divide the deployment area into disjoint geographic regions. Each robot is assigned to a unique region. Then they may shrink or expand their local regions by embracing damaged nodes or passive sensors from adjacent regions. This depends on whether they are short of spare units or are asked to help in the replacement of faulty units in nearby areas, and whether the change would benefit both regions according to some objective function. As the local regions are dynamically changing, the robots recalculate the replacement trajectories therein. The calculation is based on either of the centralized algorithms proposed in Chapter 4. The objective function modeling the interactions between robotic teammates must certainly be related to the sensor distributions in their corresponding regions and the length of the ensuing tours. Multiple objective functions with different purposes may be tried. This approach is localized in the sense that each robot only needs to exchange information with those in adjacent regions. No global knowledge is required. This solution is different from the randomized algorithm in [57], as it is deterministic and we hope also it has some sort of performance guarantee. The novelty lies in both the problem formulation and the localized solution itself, as there are no previous works in the literature.

8.2.3 On Fault Identification Frameworks

The extension of the search-driven fault identification process in diagnosable systems to other test models (e.g. the broadcast model) is also of a practical relevance.

Additionally, an innovative distributed diagnosis framework that no longer relies on binary test outcomes but on numerical test estimates of the extent of the damage associated with a particular node is highly appealing to distributed computing researchers. Such a system bears a superior interpretability and may lead to more accurate results.

In the risk management framework arena, future research efforts are to focus on developing

an incremental clustering architecture, in which the feature vectors arrive one by one at the base station in an asynchronous fashion. An empirical comparison with the present snapshot-based clustering scheme shall disclose interesting insights on the variability of the structural patterns undergone by the WSN over time. This project is presently ongoing.

BIBLIOGRAPHY

[1] M. H. Afshar. Partially Constrained Ant Colony Optimization Algorithm for the Solution of Constrained Optimization Problems: Application to Storm Water Network Design. *Advances in Water Resources*, 30(4):954–965, 2007.

[2] R. Ahlswede and H. Aydinian. On Diagnosability of Large Multiprocessor Networks. *Discrete Applied Mathematics*, 156:3464–3474, November 2008.

[3] P. Angelov, D. P. Filev, and N. Kasabov. *Evolving Intelligent Systems: Methodology and Applications*. John Wiley & Sons, 2010.

[4] S. Anily and J. Bramel. Approximation Algorithms for the Capacitated Traveling Salesman Problem with Pickups and Deliveries. *Naval Research Logistics*, 46(6):654–670, 1999.

[5] C. Archetti, D. Feillet, A. Hertz, and M. G. Speranza. The Capacitated Team Orienteering and Profitable Tour Problems. *Journal of the Operational Research Society*, 60(6):831–842, 2009.

[6] C. Archetti, A. Hertz, and M. G. Speranza. Metaheuristics for the Team Orienteering Problem. *Journal of Heuristics*, 13:49–76, February 2007.

[7] J. Aubert, T. Schaberreiter, C. Incoul, D. Khadraoui, and B. Gateau. Risk-Based Methodology for Real-Time Security Monitoring of Interdependent Services in Critical Infrastructures. In *Int'l Conference on Availability, Reliability and Security*, pages 262–267, February 2010.

[8] G. Ausiello, V. Bonifaci, and L. Laura. The Online Prize-Collecting Traveling Salesman Problem. *Information Processing Letters*, 107:199–204, August 2008.

[9] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala. New Approximation Guarantees for Minimum-Weight K-Trees and Prize-Collecting Salesmen. *SIAM Journal of Computing*, 28:254–262, February 1999.

[10] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T. H. Lai. Deploying Wireless Sensors to Achieve Both Coverage and Connectivity. In *Proceedings of the 7th ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '06, pages 131–142, New York, NY, USA, 2006. ACM.

[11] E. Balas. The Prize Collecting Traveling Salesman Problem. *Networks*, 19(6):621–636, 1989.

[12] N. Bartolini, T. Calamoneri, E. G. Fusco, A. Massini, and S. Silvestri. Push & Pull: Autonomous Deployment of Mobile Sensors for a Complete Coverage. *Wireless Networks*, 16:607–625, April 2010.

[13] R. D. Baruah and P. Angelov. Evolving Fuzzy Systems for Data Streams: a Survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(6):461–476, 2011.

[14] M. A. Batalin and G. S. Sukhatme. The Analysis of an Efficient Algorithm for Robot Coverage and Exploration based on Sensor Network Deployment. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA))*, page 34783485, Barcelona, Spain, 2005.

[15] J.-F. Bérubé, M. Gendreau, and J.-Y. Potvin. A Branch-and-Cut Algorithm for the Undirected Prize Collecting Traveling Salesman Problem. *Networks*, 54:56–67, August 2009.

[16] M. Bhushan and R. Rengaswamy. Comprehensive Design of a Sensor Network for Chemical Plants Based on Various Diagnosability and Reliability Criteria. 1. Framework. *Industrial & Engineering Chemistry Research*, 41(7):1826–1839, 2002.

[17] D. M. Blough and A. Pelc. Complexity of fault diagnosis in comparison models. *IEEE Transactions on Computers*, 41:318–324, March 1992.

[18] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Inc., New York, NY, USA, 1999.

[19] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. *Wireless Networks*, 7(6):609–616, 2001.

[20] D. Bratton and J. Kennedy. Defining a Standard for Particle Swarm Optimization. In *Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS)*, pages 120 – 127, Honolulu, Hawaii, USA, April 2007.

[21] D. Calitoiu and D. Milici. Modeling with Non-Cooperative Agents: Destructive and Non-Destructive Search Algorithms for Randomly Located Objects. In N. Mansour, editor, *Search Algorithms and Applications*, Numerical Analysis and Scientific Computing. InTech, 2010.

[22] A. Casteigts, J. Albert, S. Chaumette, A. Nayak, and I. Stojmenovic. Biconnecting a Network of Mobile Robots Using Virtual Angular Forces. In *Proceedings of the 72nd IEEE Vehicular Technology Conference (VTC2010-Fall)*, pages 1–5, Ottawa, Canada, 2010.

[23] R. Cavill, S. L. Smith, and A. M. Tyrrell. Variable-length genetic algorithms with multiple chromosomes on a variant of the onemax problem. In *Proceedings of the*

8th Annual Conference on Genetic and Evolutionary Computation, GECCO '06, pages 1405–1406, New York, NY, USA, 2006. ACM.

[24] P. Chalasani and R. Motwani. Approximating Capacitated Routing and Delivery Problems. *SIAM Journal of Computing*, 28:2133–2149, August 1999.

[25] C.-Y. Chang, C.-T. Chang, Y.-C. Chen, and H.-R. Chang. Obstacle-Resistant Deployment Algorithms for Wireless Sensor Networks. *IEEE Transactions on Vehicular Technology*, 58(6):2925–2941, 2009.

[26] C.-Y. Chang, J.-P. Sheu, Y.-C. Chen, and S.-W. Chang. An Obstacle-Free and Power-Efficient Deployment Algorithm for Wireless Sensor Networks. *IEEE Transactions on Systems, Man & Cybernetics, Part A*, 39:795–806, July 2009.

[27] I.-M. Chao, B. L. Golden, and E. A. Wasil. The Team Orienteering Problem. *European Journal of Operational Research*, 88(3):464 – 474, 1996.

[28] A. Chaves and L. Lorena. Hybrid Metaheuristic for the Prize Collecting Travelling Salesman Problem. In J. van Hemert and C. Cotta, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 4972 of *Lecture Notes in Computer Science*, pages 123–134. Springer Berlin / Heidelberg, 2008.

[29] J. Chen, S. Li, and Y. Sun. Novel Deployment Schemes for Mobile Sensor Networks. *Sensors s*, 7(11):2907–2919, 2007.

[30] M. X. Cheng, L. Ruan, and W. Wu. Coverage Breach Problems in Bandwidth-constrained Sensor Networks. *ACM Transactions on Sensor Networks*, 3(2):89–124, 2007.

[31] K.-Y. Chwa and S. L. Hakimi. Schemes for Fault-Tolerant Computing: a Comparison of Modularly Redundant and t-Diagnosable Systems. *Information and Control*, 49(3):212 – 238, 1981.

[32] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Saripalli, and G. Sukhatme. Deployment and Connectivity Repair of a Sensor Net with a Flying Robot. In M. Ang and O. Khatib, editors, *Experimental Robotics IX*, volume 21 of *Springer Tracts in Advanced Robotics*, pages 333–343. Springer Berlin / Heidelberg, 2006.

[33] N. Correll, N. Arechiga, A. Bolger, M. Bollini, B. Charrow, A. Clayton, F. Dominguez, K. Donahue, S. Dyar, L. Johnson, H. Liu, A. Patrikalakis, T. Robertson, J. Smith, D. Soltero, M. Tanner, L. White, and D. Rus. Building a Distributed Robot Garden. In *Proceedings of the 2009 International Conference on Intelligent Robots and Systems (IROS)*, pages 1509–1516, St. Louis, Missouri, 2009.

[34] K. De-Jong. *Evolutionary Computation: a Unified Approach*. MIT Press, 2007.

[35] M. A. M. de Oca, T. Stützle, M. Birattari, and M. Dorigo. Frankenstein's PSO: A Composite Particle Swarm Optimization Algorithm. *IEEE Transactions on Evolutionary Computation*, 13:1120–1132, October 2009.

[36] M. Dell'Amico, F. Maffioli, and A. Sciomachen. A Lagrangian Heuristic for the Prize Collecting Travelling Salesman Problem. *Annals of Operations Research*, 81:289–306, 1998.

[37] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, December 2006.

[38] M. Dorigo and L. M. Gambardella. Ant Colonies for the Traveling Salesman Problem. *Biosystems*, 43:73–81, 1997.

[39] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, 2004.

[40] M. Dror, D. Fortin, and C. Roucairol. Redistribution of Self-Service Electric Cars: a Case of Pickup and Delivery. Technical Report RR-3543, INRIA-Rocquencourt, 1998.

[41] M. Dunbabin, I. Vasilescu, P. Corke, and D. Rus. Data Muling over Underwater Wireless Sensor Networks using an Autonomous Underwater Vehicle. In *Proceedings of the 2006 Int'l Conference on Robotics and Automation*, pages 2091–2098, Orlando, FL, 2006.

[42] M. Elhadef and B. Ayeb. An Evolutionary Algorithm for Identifying Faults in t-Diagnosable Systems. In *Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems*, pages 74–83, Washington, DC, USA, 2000. IEEE Computer Society.

[43] M. Elhadef, S. Das, and A. Nayak. System-Level Fault Diagnosis Using Comparison Models: an Artificial-Immune-Systems-Based Approach. *Journal of Networks*, 1(5):43–53, 2006.

[44] M. Elhadef, A. Nayak, and N. Zeng. An Ant-based Fault Identification Algorithm for Distributed and Parallel Systems. In *Proceedings of the 10th World Conference on Integrated Design & Process Technology (IDPT)*, pages 1–6, Antalya, Turkey, June 2007.

[45] R. Falcon, R. Abielmona, and A. Nayak. An Evolving Risk Management Framework for Wireless Sensor Networks. In *Proceedings of the 2011 IEEE Int'l Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA)*, pages 1–6, Ottawa, Canada, September 2011.

[46] R. Falcon, M. Almeida, and A. Nayak. A Binary Particle Swarm Optimization Approach to Fault Diagnosis in Parallel and Distributed Systems. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC)*, pages 41–48, Barcelona, Spain, 2010.

[47] R. Falcon, M. Almeida, and A. Nayak. Fault Identification with Binary Adaptive Fireflies in Parallel and Distributed Systems. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1359–1366, New Orleans, Louisiana, June 2011.

[48] R. Falcon, B. Depaire, and A. Caris. The One-Commodity Traveling Salesman Problem with Selective Pickup and Delivery. In *Proceedings of the 25th Conference of the Belgian Operations Research Society (ORBEL 25)*, Ghent, Belgium, 2011.

[49] R. Falcon, B. Depaire, A. Caris, X. Li, K. Vanhoof, A. Nayak, and I. Stojmenovic. A Hybrid Metaheuristic to the Single-Carrier Coverage Repair Problem in Wireless Sensor and Robot Networks. *A manuscript submitted to IEEE Trans. on Evolutionary Computation.*

[50] R. Falcon, X. Li, and A. Nayak. Carrier-Based Coverage Augmentation in Wireless Sensor and Robot Networks. In *Proceedings of the IEEE 30th International Conference on Distributed Computing Systems Workshops*, pages 234–239, Washington, DC, USA, 2010. IEEE Computer Society.

[51] R. Falcon, X. Li, and A. Nayak. Carrier-Based Focused Coverage Formation in Wireless Sensor and Robot Networks. *IEEE Transactions on Automatic Control*, 56(10):2406–2417, October 2011.

[52] R. Falcon, X. Li, A. Nayak, and I. Stojmenovic. The One-Commodity Traveling Salesman Problem with Selective Pickup and Delivery: an Ant Colony Approach. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 4326–4333, Barcelona, Spain, 2010.

[53] R. Falcon, X. Li, A. Nayak, and I. Stojmenovic. A Harmony-Seeking Firefly Algorithm to the Periodic Replacement of Damaged Sensors by a Team of Mobile Robots. In *Accepted for publication at the IEEE International Conference on Communications*, Ottawa, Canada, 2012.

[54] D. Feillet, P. Dejax, and M. Gendreau. Traveling Salesman Problems with Profits. *Transportation Science*, 39:188–205, May 2005.

[55] M. Fischetti and P. Toth. An Additive Approach for the Optimal Solution of the Prize-Collecting Travelling Salesman Problem. In B. Golden and A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 319–343. Elsevier Science Publishers, 1988.

[56] G. Fletcher, X. Li, A. Nayak, and I. Stojmenovic. Back-Tracking Based Sensor Deployment by a Robot Team. In *Proceedings of the 7th Annual IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks (SECON)*, pages 1 – 9, Boston, MA, USA, June 2010.

[57] G. Fletcher, X. Li, A. Nayak, and I. Stojmenovic. Randomized Robot-assisted Relocation of Sensors for Coverage Repair in Wireless Sensor Networks. In *Proceedings of the 72nd IEEE Vehicular Technology Conference (VTC-Fall)*, pages 1 – 5, Ottawa, Canada, October 2010.

[58] A. D. Friedman and L. Simoncini. System-Level Fault Diagnosis. *Computer*, 13:47–53, March 1980.

[59] A. Gallais, J. Carle, D. Simplot-Ryl, and I. Stojmenovic. Localized Sensor Area Coverage with Low Communication Overhead. *IEEE Transactions on Mobile Computing*, 7(5):661–672, 2008.

[60] Z. W. Geem. *Music-Inspired Harmony Search Algorithm: Theory and Applications*. Springer, 1st edition, 2009.

[61] A. Gelman and J. Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, 2007.

[62] D. Goldberg and K. Sastry. *Genetic Algorithms: The Design of Innnovation*. Springer, 2007.

[63] B. L. Golden, S. Raghavan, and E. Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer Verlag, 2008.

[64] V. C. Gungor and G. P. Hancke. Industrial Wireless Sensor Networks: Challenges, Design Principles and Technical Approaches. *IEEE Transactions on Industrial Electronics*, 56(10):4258–4265, 2009.

[65] S. L. Hakimi and A. T. Amin. Characterization of Connection Assignment of Diagnosable Systems. *IEEE Transactions on Computers*, 23:86–88, January 1974.

[66] K. Haslum and A. Arnes. Multisensor Real-time Risk Assessment using Continuous-time Hidden Markov Models. In *Int'l Conference on Computational Intelligence and Security*, pages 1536–1540, Nov. 2006.

[67] M. Hatler, D. Gurganious, and C. Chi. *Wireless Sensor Networks in Oil and Gas*. On World, 2008.

[68] M. Hatler, D. Gurganious, C. Chi, and M. Ritter. *Wireless Sensor Networks for Smart Buildings*. On World, 2009.

[69] H. Hernández-Pérez. *Traveling Salesman Problems with Pickups and Deliveries*. PhD thesis, University of La Laguna, Spain, 2004.

[70] H. Hernández-Pérez, I. Rodríguez-Martín, and J. J. Salazar-González. A Hybrid GRASP/VND Heuristic for the One-Commodity Pickup-and-Delivery Traveling Salesman Problem. *Computers & Operations Research*, 36:1639–1645, May 2009.

[71] H. Hernández-Pérez and J.-J. Salazar-González. A Branch-and-Cut Algorithm for a Traveling Salesman Problem with Pickup and Delivery. *Discrete Applied Mathematics*, 145:126–139, December 2004.

[72] H. Hernández-Pérez and J.-J. Salazar-González. Heuristics for the One-Commodity Pickup-and-Delivery Traveling Salesman Problem. *Transportation Science*, 38:245–255, May 2004.

[73] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[74] J. Hox. *Multilevel Analysis: Techniques and Applications*. Lawrence Erlbaum Associates, 2002.

[75] R. Iyer and L. Kleinrock. QoS Control for Sensor Networks. In *Proc. of the IEEE International Conference on Communications*, volume 1, pages 517 – 521 vol.1, May 2003.

[76] H. Jiang, Y. Liu, and L. Zheng. Design and Simulation of Simulated Annealing Algorithm with Harmony Search. In Y. Tan, Y. Shi, and K. Tan, editors, *Advances in Swarm Intelligence*, volume 6146 of *Lecture Notes in Computer Science*, pages 454–460. Springer Berlin / Heidelberg, 2010.

[77] T. Kameda, S. Toida, and F. Allan. A Diagnosis Algorithm for Networks. *Information & Control*, 29:141–148, 1975.

[78] J. Kennedy and R. C. Eberhart. Particle Swarm Optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, ICNN '95, pages 1942–1948, 1995.

[79] J. Kennedy and R. C. Eberhart. A Discrete Binary Version of the Particle Swarm Algorithm. In *Proceedings of the 1997 Int'l Conference on Systems, Man and Cybernetics (SMC)*, pages 4104–4108, Orlando, Florida, USA, October 1997.

[80] S. E. Kreutzer and S. L. Hakimi. System-Level Diagnosis: Analysis of Two New Models. *Information Sciences*, 40:117–130, December 1986.

[81] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic. Genetic Algorithms for the Travelling Salesman Problem: a Review of Representations and Operators. *Artificial Intelligence Review*, 13:129–170, April 1999.

[82] K. Latha and R. Manivelu. Music-Inspired Optimization Algorithm: Harmony-Tabu for Document Retrieval Using Relevance Feedback. In V. V. Das, R. Vijayakumar, N. C. Debnath, J. Stephen, N. Meghanathan, S. Sankaranarayanan, P. M. Thankachan, F. L. Gaol, and N. Thankachan, editors, *Information Processing and Management*, volume 70 of *Communications in Computer and Information Science*, pages 385–387. Springer Berlin Heidelberg, 2010.

[83] M. Lazarova and P. Borovska. Comparison of Parallel Metaheuristics for Solving the TSP. In *Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*, CompSysTech '08, pages 17:II.12–17:1, New York, NY, USA, 2008. ACM.

[84] T. Le, N. Ahmed, and S. Jha. Location-Free Fault Repair in Hybrid Sensor Networks. In *Proceedings of the 1st Int'l Conference on Integrated Internet Ad Hoc and Sensor Networks*, InterSense '06, New York, NY, USA, 2006. ACM.

[85] W. Li, Y. I. Kamil, and A. Manikas. Wireless Array based Sensor Relocation in Mobile Sensor Networks. In *Proceedings of the 2009 Int'l Conference on Wireless Communications and Mobile Computing*, IWCMC '09, pages 832–838, New York, NY, USA, 2009. ACM.

[86] X. Li. *Improving Area Coverage by Mobile Sensor Networks*. PhD thesis, Carleton University, Ottawa, Canada, 2008.

[87] X. Li, H. Frey, N. Santoro, and I. Stojmenovic. Localized Self-Deployment of Mobile Sensors for Optimal Focused-Coverage Formation. Technical Report TR-2007-13, SITE, University of Ottawa, December 2007.

[88] X. Li, H. Frey, N. Santoro, and I. Stojmenovic. Focused Coverage by Mobile Sensor Networks. In *Proceedings of the 6th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pages 466–475, Macau, China, 2009.

[89] X. Li, H. Frey, N. Santoro, and I. Stojmenovic. Strictly Localized Sensor Self-Deployment for Optimal Focused Coverage. *IEEE Transactions on Mobile Computing*, 10(11):1520–1533, November 2011.

[90] X. Li, N. Mitton, I. Ryl, and D. Simplot. Localized Sensor Self-Deployment with Coverage Guarantee in Complex Environment. In *Proceedings of the 8th International Conference on AD-HOC Networks & Wireless (AdHoc-Now) LNCS 5793*, pages 138–151, Murcia, Spain, 2009.

[91] X. Li, A. Nayak, and I. Stojmenovic. Sensor Placement in Sensor and Actuator Networks. In A. Nayak and I. Stojmenovic, editors, *Wireless Sensor and Actuator Networks. Algorithms and Protocols for Scalable Coordination and Data Communication*, chapter 10. Wiley VCH, 2010.

[92] X. Li and N. Santoro. An Integrated Self-deployment and Coverage Maintenance Scheme for Mobile Sensor Networks. In J. Cao, I. Stojmenovic, X. Jia, and S. Das, editors, *Mobile Ad-hoc and Sensor Networks*, volume 4325 of *Lecture Notes in Computer Science*, pages 847–860. Springer Berlin / Heidelberg, 2006.

[93] X. Li and N. Santoro. ZONER: A ZONE-based Sensor Relocation Protocol for Mobile Sensor Networks. In *Proceedings of the 31st IEEE Conference on Local Computer Networks*, pages 923–930, Tampa, FL, USA, November 2006.

[94] X. Li, N. Santoro, and I. Stojmenovic. Mesh-Based Sensor Relocation for Coverage Maintenance in Mobile Sensor Networks. In J. Indulska, J. Ma, L. Yang, T. Ungerer, and J. Cao, editors, *Ubiquitous Intelligence and Computing*, volume 4611 of *Lecture Notes in Computer Science*, pages 696–708. Springer Berlin / Heidelberg, 2007.

[95] A. Ligeza. *Logical Foundations for Rule-Based Systems*, volume 11 of *Studies in Computational Intelligence*. Springer Verlag, 2006.

[96] H. Liu, X. Chu, Y.-W. Leung, and R. Du. Simple Movement Control Algorithm for Bi-Connectivity in Robotic Sensor Networks. *IEEE Journal on Selected Areas in Communications*, 28(7):994–1005, September 2010.

[97] L. Lopez, M. W. Carter, and M. Gendreau. The Hot Strip Mill Production Scheduling Problem: a Tabu Search Approach. *European Journal of Operational Research*, 106(2-3):317–335, 1998.

[98] M. Ma and Y. Yang. Adaptive Triangular Deployment Algorithm for Unattended Mobile Sensor Networks. *IEEE Transactions on Computers*, 56:946–847, July 2007.

[99] J. Maeng and M. Malek. A Comparison Connection Assignment for Self-Diagnosis of Multiprocessor Systems. In *Proceedings of the 11th Fault-Tolerant Computing Symposium*, pages 173–175, 1981.

[100] G. Martinovic, I. Aleksi, and A. Baumgartner. Single-Commodity Vehicle Routing Problem with Pickup and Delivery Service. *Mathematical Problems in Engineering*, 2008, 2008.

[101] Y. Mei, C. Xian, S. Das, Y. C. Hu, and Y.-H. Lu. Sensor Replacement Using Mobile Robots. *Computer Communications*, 30:2615–2626, September 2007.

[102] T. Melodia, D. Pompili, and I. F. Akyldiz. Handling Mobility in Wireless Sensor and Actor Networks. *IEEE Transactions on Mobile Computing*, 9:160–173, February 2010.

[103] R. Mendes, J. Kennedy, and J. Neves. The Fully Informed Particle Swarm: Simpler, Maybe Better. *IEEE Transactions on Evolutionary Computation*, 8(3):204–210, June 2004.

[104] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs, 3rd Ed.* Springer-Verlag, 1996.

[105] S. Mitra, W. Pedrycz, and B. Barman. Shadowed C-means: Integrating Fuzzy and Rough Clustering. *Pat. Recognition*, 43(4):1282 – 1291, 2010.

[106] F. Nadi, A. T. Khader, and M. A. Al-Betar. Adaptive Genetic Algorithm using Harmony Search. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 819–820, New York, NY, USA, 2010. ACM.

[107] K. S. Narendra and M. A. L. Thathachar. *Learning Automata*. Prentice Hall, 1989.

[108] A. Nayak and I. Stojmenovic. *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*. John Wiley & Sons, 2010.

[109] S. C. Negulescu, C. Oprean, C. V. Kifor, and I. Carabulea. Elitist Ant System for Route Allocation Problem. In *Proceedings of the 8th Conference on Applied Informatics and Communications*, pages 62–67, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[110] B. J. Oommen, O.-C. Granmo, and A. Pedersen. Empirical Verification of a Strategy for Unbounded Resolution in Finite Player Goore Games. In A. Sattar and B. ho Kang, editors, *AI 2006: Advances in Artificial Intelligence*, volume 4304 of *Lecture Notes in Computer Science*, pages 1252–1258. Springer Berlin / Heidelberg, 2006.

[111] B. J. Oommen, O.-C. Granmo, and A. Pedersen. Using Stochastic AI Techniques to Achieve Unbounded Resolution in Finite Player Goore Games and its Applications. In *Proc. of the IEEE Symposium on Computational Intelligence and Games*, pages 161 – 167, April 2007.

[112] I. Or. *Traveling Salesman-type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking*. PhD thesis, Northwestern University, Evanston, IL, 1976.

[113] S. Parragh, K. Doerner, and R. Hartl. A Survey on Pickup and Delivery Problems, Part I: Transportation Between Customers and Depot. *Journal für Betriebswirtschaft*, 58:21–51, 2008.

[114] S. Parragh, K. Doerner, and R. Hartl. A Survey on Pickup and Delivery Problems, Part II: Transportation Between Pickup and Delivery Locations. *Journal für Betriebswirtschaft*, 58:81–117, 2008.

[115] W. Pedrycz. Shadowed Sets: Bridging Fuzzy and Rough Sets. In S. K. Pal and A. Skowron, editors, *Rough Fuzzy Hybridization. A New Trend in Decision-Making*, pages 179–199. Springer Verlag / Singapore, 1999.

[116] W. Pedrycz. *Granular Computing: an Emerging Paradigm*, volume 70 of *Studies in Fuzziness and Soft Computing*. Springer, 2001.

[117] W. Pedrycz. From Fuzzy Sets to Shadowed Sets: Interpretation and Computing. *Int'l Journal of Intelligent Systems*, 24:48–61, 2009.

[118] W. Pedrycz. Evolvable Fuzzy Systems: Some Insights and Challenges. *Evolving Systems*, 1:73–82, 2010.

[119] A. Pelc. Undirected Graph Models for System-Level Fault Diagnosis. *IEEE Transactions on Computers*, 40:1271–1276, November 1991.

[120] F. P. Preparata, G. Metze, and R. T. Chien. On the Connection Assignment of Diagnosable Systems. *IEEE Transactions on Electronic Computing*, 16:848–854, December 1967.

[121] A. Puris. *Desarrollo de Metaheurísticas Poblacionales para la Solución de Problemas Complejos*. PhD thesis, Universidad Central de Las Villas, Santa Clara, Cuba, 2010. (in Spanish).

[122] A. Puris, R. Bello, Y. Trujillo, A. Nowe, and Y. Martínez. Two-Stage ACO to Solve the Job Shop Scheduling Problem. In *Proceedings of the 12th Iberoamerican Conference on Pattern Recognition, Image Analysis and Applications*, CIARP'07, pages 447–456, Berlin, Heidelberg, 2007. Springer-Verlag.

[123] A. K. Qin and F. Forbes. Harmony Search with Differential Mutation based Pitch Adjustment. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 545–552, New York, NY, USA, 2011. ACM.

[124] G. Reinelt. *The Traveling Salesman Problem: Computational Solutions for TSP Applications*. Berlin Springer-Verlag, 1994.

[125] V. Rodoplu and T. H. Meng. Minimum Energy Mobile Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 17:1333–1344, August 1999.

[126] M. K. Sayadi, R. Ramezanian, and N. Ghaffari-Nasab. A Discrete Firefly Meta-Heuristic with Local Search for Makespan Minimization in Permutation Flow Shop Scheduling Problems. *Int'l Journal of Industrial Engineering Computations*, 1:1–10, April 2010.

[127] T. Schaberreiter, C. Bonhomme, J. Aubert, C. Incoul, and D. Khadraoui. Support Tool Development for Real-Time Risk Prediction in Interdependent Critical Infrastructures. In *IEEE Int'l Symposium on Sofware Reliability Engineering*. 2009.

[128] L. C. Shiu. The Robot Deployment Scheme for Wireless Sensor Networks in the Concave Region. In *Proceedings of the 2009 Int'l Conference on Networking, Sensing and Control*, pages 581–586, Okayama, Japan, March 2009.

[129] H. Shwe, B. Lee, and J.-S. Lee. DoS Attack Mining in Sensor Node Replacement. In P. Thulasiraman, X. He, T. Xu, M. Denko, R. Thulasiram, and L. Yang, editors, *Frontiers of High Performance Computing and Networking ISPA 2007 Workshops*, volume

4743 of *Lecture Notes in Computer Science*, pages 11–19. Springer Berlin / Heidelberg, 2007.

[130] K. Sohraby, D. Minoli, and T. F. Znati. *Wireless Sensor Networks: Technologies, Protocols and Applications*. Wiley Interscience, 2007.

[131] T. Stützle and H. H. Hoos. MAX-MIN Ant System. *Future Generation Computer Systems*, 16:889–914, June 2000.

[132] G. F. Sullivan. A O(t^3 + |E|) Fault Identification Algorithm for Diagnosable Systems. *IEEE Transactions on Computers*, 37:388–397, April 1988.

[133] X. Tan, Y. Zhang, X. Cui, and H. Xi. Using Hidden Markov Models to Evaluate the Real-Time Risks of Network. In *IEEE Int'l Symposium on Knowledge Acquisition and Modeling Workshop*, pages 490–493, December 2008.

[134] W. J. Tang and Q. H. Wu. Biologically Inspired Optimization: a Review. *Transactions of the Institute of Measurement and Control*, 31:495–515, 2009.

[135] J. Tavares, F. J. Velez, and J. M. Ferro. Application of Wireless Sensor Networks to Automobiles. *Measurement Science Review*, 8(3):65–70, 2008.

[136] M. L. Tsetlin. *Automaton Theory and Modeling of Biological Systems*. Academic Press, New York and London, 1973.

[137] B. Tung and L. Kleinrock. Using Finite State Automata to Produce Self-Optimization and Self-Control. *IEEE Transactions on Parallel and Distributed Systems*, 7(4):439–448, Apr. 1996.

[138] K. Tzu-Liang, C. Hsing-Chung, and J. Tan. On the Faulty Sensor Identification Algorithm of Wireless Sensor Networks under the PMC Diagnosis Model. In *Networked Computing and Advanced Information Management (NCM), 2010 Sixth International Conference on*, pages 657–661, August 2010.

[139] J. Valente de Oliveira and W. Pedrycz. *Advances in Fuzzy Clustering and its Applications*. John Wiley & Sons, 2007.

[140] C. A. Valle, L. C. Martinez, A. S. da Cunha, and G. R. Mateus. Heuristic and Exact Algorithms for a Min-Max Selective Vehicle Routing Problem. *Computers & Operations Research*, 38:1054–1065, July 2011.

[141] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A Break in the Clouds: Towards a Cloud Definition. *SIGCOMM Computer Communication Review*, 39:50–55, December 2008.

[142] G. Wang, G. Cao, and T. L. Porta. Proxy-Based Sensor Deployment for Mobile Sensor Networks. In *Proceedings of the 2004 IEEE Int'l Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pages 493–502, Fort Lauderdale, FL, USA, 2004.

[143] G. Wang, G. Cao, T. L. Porta, and W. Zhang. Sensor Relocation in Mobile Sensor Networks. In *Proceedings of the 24th Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM)*, pages 2302 –2312 vol. 4, Miami, FL, USA, 2005.

[144] T. White and A. Salehi-Abari. A Swarm-Based Crossover Operator for Genetic Programming. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, pages 1345–1346, New York, NY, USA, 2008. ACM.

[145] J. Wu and Z. Jiang. A Hierarchical Structure based Coverage Repair In Wireless Sensor Networks. In *Proceedings of the 19th Int'l Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–6, Cannes, France, 2008.

[146] H. Yang, M. Elhadef, A. Nayak, and X. Yang. Network Fault Diagnosis: An Artificial Immune System Approach. In *Proc. of the 14th IEEE Int'l Conference on Parallel and Distributed Systems*, pages 463 – 469, Melbourne, Australia, December 2008.

[147] H. Yang, X. Yang, and A. Nayak. A (4n - 9)/3 diagnosis algorithm for generalised cube networks. *Int. J. Parallel Emerg. Distrib. Syst.*, 25:171–182, June 2010.

[148] X. Yang, G. M. Megson, and D. J. Evans. A Comparison-based Diagnosis Algorithm Tailored for Crossed Cube Multiprocessor Systems. *Microprocessors and Microsystems*, 29(4):169 – 175, 2005.

[149] X.-S. Yang. Firefly Algorithms for Multimodal Optimization. In *Proc. of the 5th Symposium on Stochastic Algorithms, Foundations and Applications (SAGA), LNCS 5792*, pages 169–178, Sapporo, Japan, 2009.

[150] X.-S. Yang. *Nature-Inspired Metaheuristic Algorithms, 2nd Ed.* Luniver Press, 2010.

[151] L. Yong-qiang, Z. Xiao-shu, and X. Miao. Improved Chromosome Encoding for Genetic-Algorithm-based Assembly Sequence Planning. In *Proceedings of the 2010 Int'l Conference on Intelligent Computing and Integrated Systems (ICISS)*, pages 35 – 39, Guiling, China, October 2010.

[152] S. F. Yuan and F. L. Chu. Fault Diagnostics based on Particle Swarm Optimisation and Support Vector Machines. *Mechanical Systems and Signal Processing*, 21(4):1787 – 1798, 2007.

[153] H. Zhang and J. C. Hou. Maintaining Sensing Coverage and Connectivity in Large Sensor Networks. *Ad Hoc & Sensor Wireless Networks*, 1:89–124, March 2005.

[154] F. Zhao, S. Li, J. Sun, and D. Mei. Genetic Algorithm for the One-commodity Pickup-and-Delivery Traveling Salesman Problem. *Computers & Industrial Engineering*, 56:1642–1648, May 2009.

[155] F. Zhao, F. Zhao, T. Li, and D. A. Bryant. A New Pheromone-Trail-Based Genetic Algorithm for Comparative Genome Assembly. *Nucleic Acids Research*, 36:3455–3462, April 2008.

[156] J. Zhou, W. Pedrycz, and D. Miao. Shadowed Sets in the Characterization of Rough-Fuzzy Clustering. *Pattern Recognition*, 44:1738–1749, August 2011.