

Reliable Robot-Assisted Sensor Relocation via Multi-Objective Optimization

Benjamin Desjardins

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Master of Computer Science degree

Ottawa-Carleton Institute of Computer Science
School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa
Ottawa, Canada

Sometimes the longer path
is the one worth taking

Abstract

Wireless sensor networks (WSNs) are an emerging area of technology that have applications across many domains. By adding a mobile platform to the WSN we can increase its capabilities. One such scenario involves a mobile platform relocating sensors to fill sensing holes that are the result of sensor failure. We examine this problem, known as robot-assisted sensor relocation (RASR), and propose our own, multi-objective version, that we call reliable robot-assisted sensor relocation. We solve this problem using a set of state-of-the-art evolutionary multi-objective optimization algorithms. Additionally, we examine the multi-robot model, which we christen reliable multiple robot-assisted sensor relocation (RMRASR). The works collected within define these problems as well as provide empirical insight into the performance of well-known algorithms using these problems as a test-bed.

Acknowledgements

My deepest thanks and regards go to my thesis supervisor, Dr. Emil Petriu, and co-supervisor, Dr. Rami Abielmona, for their support and guidance throughout my tenure in the masters program. They have provided me many opportunities to expand both my academic and personal horizons.

I would also like to thank Dr. Rafael Falcon, who has introduced me to the world of academic research and guided me along this path. Without your experience, patience, and advice through many meetings and discussions I would not be where I am today. It has been an honour to work alongside you these past two years.

Special thanks go out to the members of Larus Technologies Corporation who have put up with me in their midst. The support and opportunities that they presented me with did not go unnoticed.

To my family and friends, thank you so much for your encouragement and patience throughout this. You kept me sane through the hard times and reminded me to stay grounded.

Finally, to my wife Justine, putting up with me working this hard must have been an ordeal, so thank you for sticking by me.

This work was partially supported by NSERC Strategic Grants from the thesis supervisor, the MITACS Accelerate program, and Larus Technologies Corporation.

Contributions

The following list enumerates the author's contribution to each of the works contained within this thesis:

90% - *A Multi-Objective Optimization Approach to Reliable Robot-Assisted Sensor Relocation* [1]

90% - *Planning Robust Sensor Relocation Trajectories for a Mobile Robot with Evolutionary Multi-Objective Optimization* [2]

90% - *Reliable Multiple Robot-Assisted Sensor Relocation using Multi-Objective Optimization* [3]

Table of Contents

Abstract	iii
Acknowledgements	iv
Contributions	v
Table of Contents	vi
List of Figures	viii
1 Introduction	1
1.1 Motivation	1
1.2 Contributions	5
1.3 Outline	6
2 Related Work	8
2.1 Wireless Sensor Networks	8
2.1.1 Sensor Deployment	9
2.1.2 Sensor Relocation	10
2.2 Optimization Problems	12
2.2.1 Complexity Theory	13
2.2.2 Travelling Salesman Problem	14
2.2.3 Vehicle Routing Problem	16
2.3 Evolutionary Multi-Objective Optimization Algorithms	17
2.3.1 Multi-Objective Optimization Algorithms	18
2.3.2 Genetic Algorithms	20
2.4 Risk-Aware Systems	25

3	A Multi-Objective Optimization Approach to Reliable Robot-Assisted Sensor Relocation	27
3.1	Preamble	27
3.2	Copyright Notice	29
3.3	A Multi-Objective Optimization Approach to Reliable Robot-Assisted Sensor Relocation	30
4	Planning Robust Sensor Relocation Trajectories for a Mobile Robot with Evolutionary Multi-Objective Optimization	43
4.1	Preamble	43
4.2	Copyright Notice	45
4.3	Planning Robust Sensor Relocation Trajectories for a Mobile Robot with Evolutionary Multi-Objective Optimization	47
5	Reliable Multiple Robot-Assisted Sensor Relocation using Multi-Objective Optimization	77
5.1	Preamble	77
5.2	Copyright Notice	79
5.3	Reliable Multiple Robot-Assisted Sensor Relocation using Multi-Objective Optimization	80
6	Conclusion	94
6.1	Discussion	94
6.2	Future Research Directions	97
7	References	98

List of Figures

1	A sensor relocation trajectory. The red nodes indicate sensing holes, the white nodes indicate passive sensors, and the green node indicates the base station where the robot begins and ends the relocation trajectory.	3
2	Relationship of algorithm complexity classes	14
3	An example of a Travelling Salesman Problem tour.	15
4	An example of a Vehicle Routing Problem solution. The coloured tours represent disjoint subtours.	16
5	Pareto set of solutions for a theoretical cost-comfort car purchasing scenario.	18
6	The flow of a typical genetic algorithm	21
7	An example of binary tournament selection for determining the parents of potential offspring solutions	21
8	An example of Partially Mapped Crossover.	22
9	An example of Cycle Crossover.	23
10	An example of Order Crossover.	23

Introduction

The purpose of this introductory chapter is to present the reader with an overview of *robot-assisted sensor relocation*, a method for maintaining the monitoring capabilities of a wireless sensor network using mobile robot platforms. We examine previous common solutions to this problem and propose a new version that is more suitable for a potential real-world application. We provide efficient solutions to this new version and examine both the solutions and their results from various angles.

After providing the user with necessary context and the research goals contained within, we present the three publications that make up the body of this work. The chapter concludes with an outline of the remainder of the thesis.

1.1 Motivation

Wireless Sensor Networks (WSNs) are an interesting area of constantly developing technology. A WSN is defined as a group of wireless sensors that are deployed in a *region*

of interest (ROI) [4]. These sensors are devices equipped with some form of monitoring hardware, a battery, and wireless communication [4]. A variety of domains require the ability to uninterruptedly monitor an ROI [4] and there is a drive to make these sensors more effective and efficient. A deployed WSN will gather low-level information about an ROI in order to be used in high-level applications. The domains in which WSNs are employed, such as defence [5], medicine [6], and agriculture [7], are often information-critical requiring the deployment of sensors to maximize and therefore the sensors must be deployed in order to maximize the coverage of an ROI. One such example is *PinPtr* [5]; a countersniper system that uses a group of deployed sensors to determine the location of an unknown shooter. Wireless sensors are placed in an ROI and contain hardware to measure time of arrival of muzzle blasts and shock waves from a fired shot. This information is passed to a base station where information fusion techniques are used to determine the origin of the shot. WSNs can be deployed across a variety of domains from systems that can increase productivity to ones that can help save lives.

Some of these networks incorporate mobile robots in order to enhance their capabilities. When a robotic element is added to a WSN it is referred to as a *Wireless Sensor and Robot Network* (WSRN) [8]. The robot element can be used to perform a variety of tasks within the network, and Falcon [9] makes the following distinction: if the task performed by the robot is not required for the operation of the WSN, it is called a *robot-assisted* WSN. Dunbabin et al. show an example of this in their system which uses an underwater mobile robot to collect data from underwater sensor nodes [10]. The alternative is when a robot performs a task integral to the functionality of the WSN; called a *robot-dependant* WSN. Sensor relocation in order to maximize network coverage is an example of this, as proposed by Yoo and Agrawal [11].

Robot-Assisted Sensor Relocation (RASR), also known as *carrier-based coverage repair* [12], is a recently formulated problem arising in a WRSN that uses robots to reconfigure the network topology as needed; this is considered a *robot-assisted* WSN. In an RASR formulation, there is an abundance of sensors deployed within the ROI; this means that only a subset of all the deployed sensors need to be active in order to provide maximum coverage. We assume that the sensors undergo some reconfiguration to form

the optimal WSN. In the event that one of these sensors' battery becomes depleted or damaged it will cause a sensing hole within the ROI. To combat this, a mobile robotic platform is also deployed in the ROI. This robot's purpose is to relocate the extra sensors, henceforth referred to as *passive sensors*, and place one in the location of each sensing hole. By replacing the damaged sensors essentially where they lay, the WSN will continue to preserve its original coverage of the ROI. The ability to replace sensors that become damaged or have their battery deplete also increases the overall lifetime and effectiveness of the network.

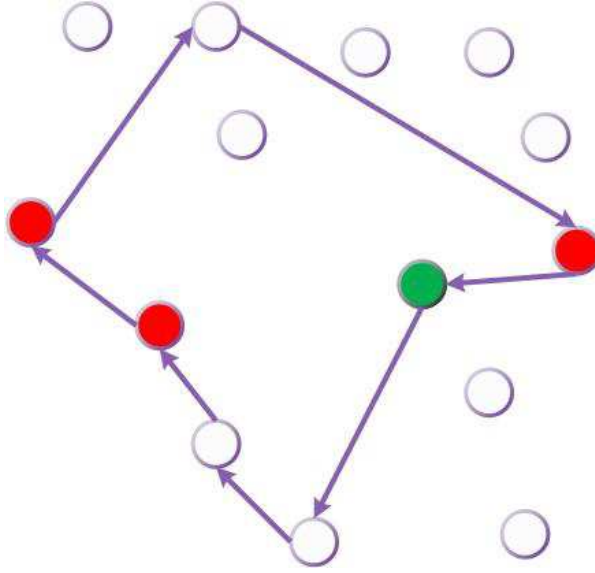


Figure 1: A sensor relocation trajectory. The red nodes indicate sensing holes, the white nodes indicate passive sensors, and the green node indicates the base station where the robot begins and ends the relocation trajectory.

In the past, the RASR problem has been algorithmically formulated as the *selective pickup and delivery problem* (SPDP) [12]. The idea is that any damaged sensors within the network need to be replaced and there is an excess of available replacements, therefore not every passive sensor needs to be visited, only the ones necessary to replace the damaged sensors. The sensor relocation is conducted by a robot that is deployed from a base station and returns there once the relocation has been completed. The optimization problem comes from trying to minimize the total distance that the robot must travel in order to replace all of the damaged sensors. This formulation is a generalization of

the classical Travelling Salesman Problem (TSP) [13] and is therefore NP-hard. A basic example of this is illustrated in Figure 1 which shows a sensor relocation trajectory. Note that while it is similar to TSP, not all nodes must be visited and the order in which they are visited matters. Also similar to the TSP, it is very difficult to compute an optimal solution for an SPDP instance and heuristics are often used to find a solution that is “good enough” and can be arrived at in a reasonable amount of time.

The issue with the approach described above is that for a solution that is considered good enough to act on, there are potential issues not taken into consideration. Passive sensors within a WSN will still use a minimum amount of battery power, therefore some will be more suitable than others for becoming replacements in order to try and preserve the coverage of the network. In WSNs, information is often transferred along a multi-hop path to reach a base station [4] and as such there are certain sensors that will end up acting as high-traffic communication nodes, especially those in the vicinity of the base station. It is essential that these sensors remain operational for as long as possible so as to not create any communication outages. If a solution is created without considering the above points, it is possible to have a WSN that is left in an unreliable state that will soon fail again. In order to provide solutions that will leave the WSN in a reliable state, the points mentioned form the basis, and inspiration, for the initial formulation of the **Reliable Robot-Assisted Sensor Relocation** (RRASR) problem.

The consideration of two additional facets of the original RASR problem, the replacement of high-traffic communication nodes with high battery level replacements and using passive sensors with the most available battery power, thrusts us into the world of multi-objective optimization (MOO). In a MOO problem, each solution must try to optimize multiple problem objectives. Generally speaking, the objectives of a MOO problem will be conflicting, thus making it very unlikely that there is a single optimal solution. There are cases where we can apply weights to the problem objectives and create what is essentially a single-objective problem, which is much easier to try and optimize. The drawback of this approach lies in the difficulty to find the exact set of weights that apply to each decision objective, hence bringing a good deal of subjectivity into the problem.

In MOO, there is a concept of *solution domination* [14] wherein one solution is said to

strongly dominate another if it performs better in all objectives than another. A solution weakly dominates another if it performs better in at least one objective and performs equal if not better in all others [15]. If a solution performs better than another in one objective but not others then they are considered as mutually non-dominated solutions.

This presents a unique problem wherein the final result of a MOO algorithm must present the user with a set of viable solutions as there will be solutions that favour one objective over the others and it is the users job to select the one most appropriate for the given situation. The algorithms used in the RRASR evaluation have been selected with this in mind in order to provide the network operator with a variety of solutions so that they may choose the best solution depending on their current situation.

1.2 Contributions

The works contained within this thesis have the consistent theme of developing the RRASR problem. The first contribution is the mathematical representation of the RRASR problem, which takes inspiration from the formulation proposed for the *selective pickup and delivery problem* [12]. This formulation, as well as all others found, is applied to a RASR problem but lacks consideration of the battery level of passive sensors as well as the placement of sensors to maintain maximum network coverage and communication. The addition of these additional objectives provides a more realistic simulation of RASR than only considering the distance travelled by the mobile robot.

The second contribution comes in the form of creating the building blocks required to solve this problem with an evolutionary multi-objective optimization (EMOO) algorithm; specifically genetic algorithms (GAs). The single-objective version of this problem has been solved prior by Liao and Ting [16] where they use a permutation that integrates positive and negative integers to represent locations that are and are not visited in a given sensor relocation trajectory, respectively. This is implemented in our proposed solution, but due to additional problem constraints there is no guarantee that a given solution will be feasible and therefore we have developed a specialized repair function to guarantee the validity of all solutions in the population and ensure that infeasible solutions are culled from the population.

Our third contribution is an additional layer to the RRASR problem; we add a risk management framework (RMF) as originally proposed by Falcon et al. [17]. The RMF that we propose is tailored specifically to the RRASR problem and enables the system to determine when sensor replacement should take place. The integration of the RMF allows the system to deploy robots when user-specified risk thresholds have been reached in order to provide the exact amount of reliability that the user desires.

The first two publications contained within this thesis provide a solution for a single-robot system. The final publication contained within provides a formulation for the multiple-robot version of the RRASR problem. The addition of multiple robots to the RRASR formulation introduced a host of problems that needed to be resolved via a change in the problem formulation to account for the multiple relocation trajectories rather than just the single one required for the original works, the addition of a new objective for load balancing, and a new repair algorithm to compensate for the new problem requirements. This is the fourth contribution of this thesis.

The final contribution is the empirical evaluation of the following state-of-the-art EMOO algorithms via the proposed formulations: NSGA-II [14], NSGA-III [18], PESA-II [19], SPEA2 [20], AGE-I [21], and AGE-II [22]. The evaluation conducted using these algorithms serves two purposes: to find which of the selected algorithms gives the best performance for RRASR, and to provide a comparative performance analysis of these algorithms outside of traditional benchmark problems.

1.3 Outline

The remainder of this the is structured as follows:

Chapter 2 presents the reader with the related background information and previously published studies that are relevant to the works contained within the thesis.

Chapter 3 presents the paper *A Multi-Objective Optimization Approach to Reliable Robot-Assisted Sensor Relocation* [1] published in the proceedings of the 2015 IEEE Congress on Evolutionary Computation.

Chapter 4 presents the chapter *Planning Robust Sensor Relocation Trajectories for a Mobile Robot with Evolutionary Multi-Objective Optimization* [2] published in the

Springer book “Computational Intelligence in Wireless Sensor Networks: Recent Advances and Future Challenges”.

Chapter 5 presents the paper *Reliable Multiple Robot-Assisted Sensor Relocation using Multi-Objective Optimization* [3] published in the proceedings of the 2016 IEEE Congress on Evolutionary Computation.

Chapter 6 concludes the thesis and details potential directions for future research.

Related Work

This chapter serves to provide a broad overview of literature pertaining the works presented within this manuscript. It examines studies related to wireless sensor and robot networks, robot-assisted sensor relocation, the underlying optimization problems, evolutionary multi-objective optimization approaches, and risk-aware systems.

2.1 Wireless Sensor Networks

Wireless sensor networks are an area of powerful emerging technology wherein a group of sensors are deployed in an ROI in order to conduct a monitoring task [4]. A wireless sensor is defined as a piece of processing hardware with a sensing device, such as a thermometer, barometer, or motion sensor; a form of wireless communication, and a power supply (e.g., battery) [4]. These networks are used in many domains, and ideally provide the maximum amount of coverage possible. Some of these WSNs use a mobile robotic platform in order to perform some sensing and/or actuation task within the network. These are referred to as WSRNs. Within WSRNs there are two specific tasks that can be represented as

optimization problems: *sensor deployment* and *sensor relocation*.

In a WSRN tasked with sensor deployment, the robot is responsible for deploying sensors in an ROI. Depending on the operational situation, this deployment can take many forms, such as: a blanket approach that attempts to entirely cover an ROI or around the perimeter of an ROI. The former is used in [23] for a wildfire detection scenario and the latter is shown in [24] for an intrusion detection scenario. In a sensor relocation scenario, the robot has the task of relocating sensors to either replace sensing gaps in the ROI due to sensor failure, when there are extra sensors, or to reconfigure the network, when there are no extra sensors.

2.1.1 Sensor Deployment

The effectiveness of a WSN depends heavily on its deployment. If a WSN is deployed without thought, then it is likely that the amount of coverage in the ROI is not maximized. In order to try and maximize the coverage area, a good deployment method must be selected. Mobile actuators, or robots, can be used to deploy a WSN and deployment methods can be divided into two groups: localized and centralized. With localized deployment, the robots use information that is available to them to determine where to place sensors. A centralized approach will determine the layout *a priori* and distribute sensors according to that layout. In some situations, an ad-hoc deployment may be required, where sensors are randomly placed, such as being dropped out of a plane. In these ad-hoc deployment, situations it is useful to have a sensor relocation method available in order to organize the nodes in a more effective manner.

There are a few types of centralized deployment methods that can be used, such as: grid [25], optimal [26], or 3-d [27] placement. Grid deployment is done by using a grid to maximize WSN coverage. An example of this can be seen in [25] where a triangular grid is used to create an optimally-spaced WSN that maximizes coverage area with minimum sensor overlap. [26] proposes an optimal deployment whereby a physical layer representation is first established and following that an optimal traffic flow is determined. Each step of the process is determined for the network in order to ensure an optimal deployment arrangement. Traffic flow within the network is considered and uses the

presence of sources and sinks within the network to determine ideal distribution in order to minimize the number of hops required for transmitting data. In [27] the authors expand on the 2-d planar deployment by adding height and line-of-sight parameters to their formulation. In doing this, they create a 3-d sensor deployment scheme in order to minimize the number of sensors required in an environment with varying elevation.

In localized methods, the information that is obtained by the robot is used to determine where to place a sensor. These methods require algorithms that allow the robot to operate independently of any controller. An example of this can be seen in [28] where a randomized approach referred to as least recently visited (LRV) is proposed. In this approach, each robot stores a weight for each direction that it travels in. This weight represents the number of instances that a robot has travelled in a given direction. The weights are then used to determine where the robot should travel next, with directions having the lowest weight given priority. [29] proposes a method where each robot aligns with a common grid and moves in a pre-defined order of preference, dropping sensors at grid points. These sensors are associated with a sequence number and are marked depending on their adjacency to an empty grid location (either white or black). The robot(s) move about the grid using this information to create coverage. This method does not require terrain and boundary information to be known *a priori* and is fault-tolerant with regards to both sensor and robot failures.

2.1.2 Sensor Relocation

Sensor relocation methods provide a way for a WSN to maintain the maximum amount of coverage possible within an ROI despite possible sensor failures [30]. Alternatively, these methods can also be used to cope with an ROI whose dimensions change over time [31]. In either case, some mobile robot must be present in order to provide the relocation service; as such, sensor relocation can be seen as a robot-dependent WSRN. In the literature, this is referred to as *robot-assisted sensor relocation* (RRSR) or *carrier-based coverage repair* [32].

Sensor relocation methods can be divided into two groups based on the situation: there are extra, unused sensors in the ROI, or the only sensors available are the ones

currently active. The method of relocation will vary based on this. In situations where all sensors are currently in use, there will be a reconfiguring of the network in order to minimize the number of sensing gaps in the ROI. In situations where there is an excess of sensors, the relocation will simply be placing one of the extra sensors in the location of failed sensors. There have been solutions proposed for both of these situations. As aforementioned, with regards to sensor deployment, sensor relocation methods can be divided into centralized and localized methods.

Centralized methods rely on some central server determining the best routing for the robots in order to relocate sensors. The robots will only follow instructions as specified by the central server. Some of these methods use optimization algorithms to create sensor relocation trajectories. An example of this is given by Falcon et al. [32], where sensor relocation is represented with *one-commodity travelling salesman problem with selective pickup and delivery* optimization. Another modified version of the travelling salesman problem is used as the basis for the approach by Miao and Yu-Ping [33] which uses a genetic algorithm as the solution method. Falcon et al. examine a multiple robot sensor replacement problem they refer to as *multiple-carrier coverage repair* (MC²R) [30]. The authors use what they term a "harmony-seeking firefly algorithm" in order to solve the problem.

Localized methods depend on the ability of the robot(s) to determine where to relocate sensors to. This type of approach relies on the robot(s) be completely autonomous. A market-based sensor relocation scheme is put forth in [34] in which sensors that are adjacent to a sensing hole will determine a representative and notify the mobile robot. This representative will tell the mobile robot which sensor it thinks is best to fill the sensing hole. The robot will then move the sensor in an attempt to restore sensor coverage to the sensing hole. [35] shows both proactive and reactive replacement methods. In both cases, there is a robot that wanders the ROI. When it detects a sensor alarm, in the reactive case, or a soon-to-fail sensor, in the proactive case, it will return to the base station to get a replacement and then replace the soon-to-fail nodes in the network. Two algorithms are proposed by Fletcher et al.: *Randomized Robot-assisted Relocation of Static Sensors* (R3S2) and a variant that uses grid placement, G-R3S2 [36]. In both

algorithms, the robot will wander the ROI until it finds a sensing hole. It will then locate a sensor that can be used as a replacement and then repair the hole. When the replacement is complete the robot then returns to randomly wandering the ROI.

2.2 Optimization Problems

Optimization is used to determine the best solution for a given problem. In these problems, we want to find the inputs to the problem that yield the best outputs. Traditionally, optimization problems can have one or many inputs and will have a single output. This is represented mathematically by the following:

$$\max f(X) \tag{1}$$

where $f(X)$ is the optimization function and X is the input set where $X = \{x_0, x_1, \dots, x_n\}$ and $x_i \in \mathbb{R}$. Depending on the problem the inputs or output may also be an integer, a permutation, a string, etc. and may be very computationally expensive to solve. The number of possible combinations on input values, known as the *input space*, determines the number of possibilities that must be evaluated to find the best output. Some problems have constraints on the inputs that help to reduce the size of the input space, but even in these cases there are still a large number of candidate solutions to evaluate. For many types of optimization problems, there are methods that aim to reduce the number of inputs that need to be tested; there are other, problem-specific, methods that find a sub-optimal yet provably good solution. Optimization problems can be further broken down into categories which can have common elements, including solution methods.

Combinatorial optimization is a type of optimization that revolves around combinations of items. A few classical examples of these are the knapsack problem [37], travelling salesman problem (TSP) [13], Chinese postman problem [38], vehicle routing problem [39], and scheduling [40]. What makes these problems similar is that, at a mathematical level, they can all be represented by graphs, i.e. can be defined as a grouping of points (vertices) and the links between them (edges). A solution for one of these problems will either be represented by one or more subsets of the graph, or a vector ordering of graph

vertices or edges. Problems of this type will have a finite number of possible solutions as there are a finite number of elements that make up a graph. In theory, it would be possible to examine each possible solution for a combinatorial optimization problem and select the best; this is known as *brute force* [41]. In practice this is generally not feasible as the number of possible combinations takes an enormous amount of time to compute [42]. Many cryptographic algorithms are based on this fact, as it would take years to crack even a single encrypted datum, such as the algorithm proposed in [42]. Problems of this kind fall into a category called NP for which there is no known method for finding an optimal solution in a reasonable amount of time but we are still able to check if a solution is valid [43].

2.2.1 Complexity Theory

All optimization problems can be placed into different groups depending on their computational complexity. The most controversial of these classes are P, NP, NP-hard, and NP-complete. Class P is the set of problems for which a solution can be found in polynomial time [44]; class NP is the set of problems for which given a solution to the problem, we can determine if it is valid in polynomial time [44]. The main argument that lies within these definitions is to determine if it is possible to equate the finding of a solution (class P) with the checking of a solution (class NP). NP-Complete problems are ones for which a set of problems for which any other problem in class NP can be transformed into in polynomial time and for which a solution can still be verified in polynomial time; this means that if a solution method for any problem in NP-complete is found, that solution method can be used on any problem in class NP [44]. NP-hard problems can be colloquially defined as “*the set of problems that is at least as tough as any other problem in NP*”. Realistically, this means any problem in NP can be transformed in polynomial time to a problem that is NP-hard, but a NP-hard problem is at least NP-complete and may not even have a polynomial time decision form [44].

Figure 2 shows the relationship between each of these problem classes. The P vs NP problem is, at the time of writing, a Millennium Problem¹ as the idea that $P \neq NP$ is

¹<http://www.claymath.org/millennium-problems/p-vs-np-problem>

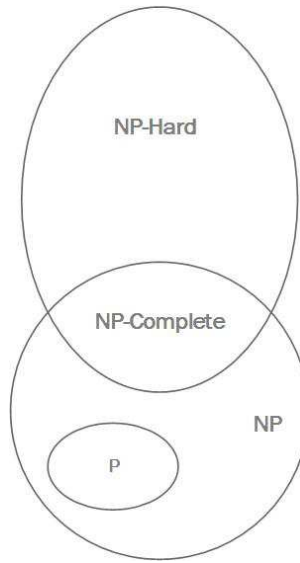


Figure 2: Relationship of algorithm complexity classes

the crux of many areas of research such as cryptography [45] and heuristics.

Just because a problem is not optimally solvable in a reasonable amount of time, that does not mean that a solution is not required for that problem. If the only scheduling solution accepted was the optimal one, we would not have transit, school, and other schedules. As such, a compromise is reached in the form of *heuristic methods* [46]. Heuristic methods allow us to approximate a so-called “good” solution to a problem. This means that while such a method may not produce the optimal solution, it will at the very least produce a valid solution. Some heuristic methods even have worst-case guarantees, such as the classical Christophedes algorithm for solving TSP [47], which guarantees the given solution will be at worst $3/2$ of the value of the optimal solution. It is methods such as these that allow us to reliably identify good solutions to problems where the optimal solution is computationally out of reach.

2.2.2 Travelling Salesman Problem

The *travelling salesman problem* is a conceptually simple problem with a NP-hard complexity. TSP is described simply as follows: if we are given a set of locations, what is the shortest distance we can travel in which we will visit all locations, travel all roads at most once, and finish the route at the starting location [13].

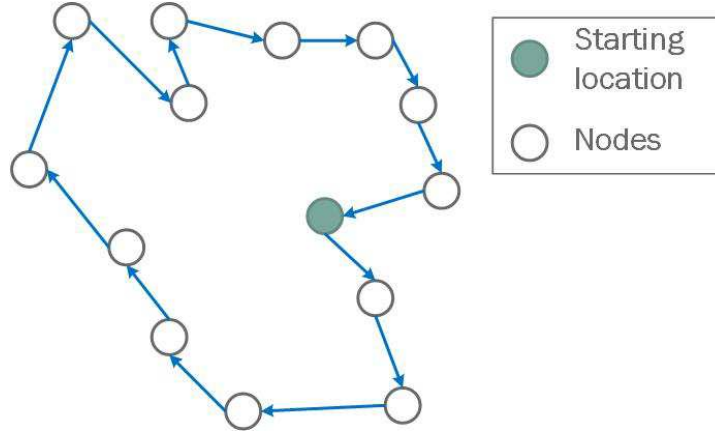


Figure 3: An example of a Travelling Salesman Problem tour.

A visual example of this is given in Figure 3. Mathematically this is represented by a graph $G = (V, E)$ where V is the set of all vertices in the graph, acting as the locations, and E is the set of all edges, acting as the routes between any two locations. The final solution to an instance of TSP is an ordered list of the vertices representing the order in which the vertices are visited.

TSP is a versatile problem that can be used as an analogue for others through the addition of various constraints. These constraints include: multiple salesman, directed edges and others [48]. These modifications allow TSP to be applied to many areas, such as: the drilling of printed circuit boards [49], cutting paper stock [50], interview scheduling [51], route planning [52], and many others.

The addition of constraints to TSP allows the formulation of other problems that have their own uses. One such problem is the *TSP with deliveries and selective pickups* (TSP-SELPD) [32]. In this problem the vertices can be divided into two subsets: pickups and deliveries. There is some commodity at each pickup location that must be collected to be delivered at the delivery locations. Each delivery location must have a commodity delivered to it, but each pickup need not be visited. This implies that a delivery location cannot be visited unless some commodity has already been collected. Additionally, there may be a constraint on how much commodity can be carried at a given time. This problem has been applied to robot-assisted sensor relocation in the past [32].

2.2.3 Vehicle Routing Problem

The vehicle routing problem (VRP) is similar to TSP in that there are one or more agents that originate at a starting location and must also end their route there. Like TSP, each location must be visited exactly once by any agent. The main difference between VRP and TSP is that in VRP it is assumed that the agent will return to the starting location more than once, therefore creating many disjoint sub-tours that all start and end with the origin [39]. As the sub-tours are disjoint, they may be served by one or more agents, depending on the number of sub-tours.

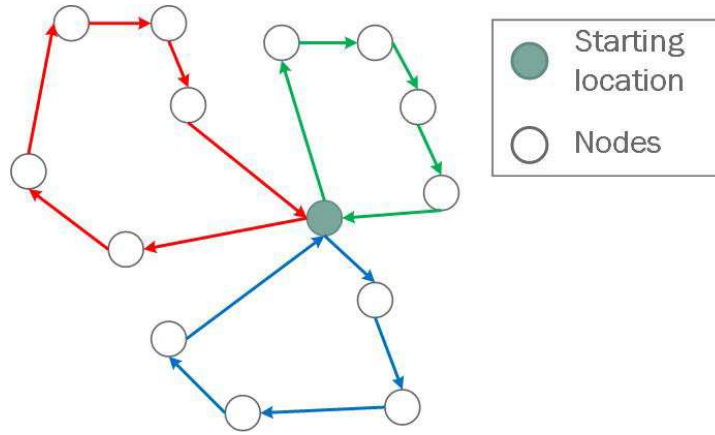


Figure 4: An example of a Vehicle Routing Problem solution. The coloured tours represent disjoint subtours.

Figure 4 gives an example of a VRP solution. The number of sub-tours can be determined by constraints put on the problem. In some cases there is a restriction on the number of agents or trips [53], in others, each agent can only carry a certain amount of commodity and must return when they are exhausted [53]; others are the reverse where the agent must pickup some commodity and when they are full they must return to the starting location, and some are a mixture of the two [54]. Other versions of the problem are constrained to time windows whereby the agents can only visit a location during certain periods of time, which may or may not be consistent across each location [55]. One version of the problem, termed *the one-commodity vehicle routing problem with selective pickup and delivery* (1-VRP-SELPD) is introduced in [56] where, similar to TSP-SELPD [32], there are a set up pickup and delivery locations wherein a delivery

must be made to all delivery locations but all pickup locations need not be visited. 1-VRP-SELPD has also been applied to RASR in the past [30].

VRP finds many real-world applications within delivery and route scheduling. It is easy to see the similarities to the planning that a distribution warehouse would do with regards to route planning and the various incarnations of VRP. A multi-depot version of VRP is proposed in [57] which is representative of multi-warehouse distribution networks used by retailers. One study, [58], expands on VRP with time-windows by modelling the transportation of perishable food which has immediate application to planning routes for food delivery.

2.3 Evolutionary Multi-Objective Optimization Algorithms

Multi-objective optimization (MOO) problems provide a new set of challenges as compared to traditional single-objective optimization problems. This stems from how solutions are determined to be superior to each other. In a single-objective problem, if a solution evaluates to be better than another it is obviously a better solution. For multi-objective problems a solution must be better in at least one objective and at least equal in all others to be considered superior. A multi-objective optimization problem can be defined as follows:

$$\max F = \{f_1(X), f_2(X), \dots, f_n(X)\} \quad (2)$$

where F is the output vector, f_i is an optimization function, and $X = \{x_0, x_1, \dots, x_n\}$ and $x_i \in \mathfrak{R}$. Each f_i may use one or many values from X . There is now an issue that must be resolved: what if a solution is better than another in one respect, but not others? I.e., what if we have solution $F_1 = \{10, 50, 100\}$ and solution $F_2 = \{100, 50, 10\}$? These solutions would be considered equally viable. The more objectives there are, the more combinations of equally viable solutions there will be. The purpose of a multi-objective algorithm is then to determine the Pareto optimal set of solutions. The latter set of solutions is the set of solutions for a given problem where the solutions cannot be improved in one objective without compromising another [59].

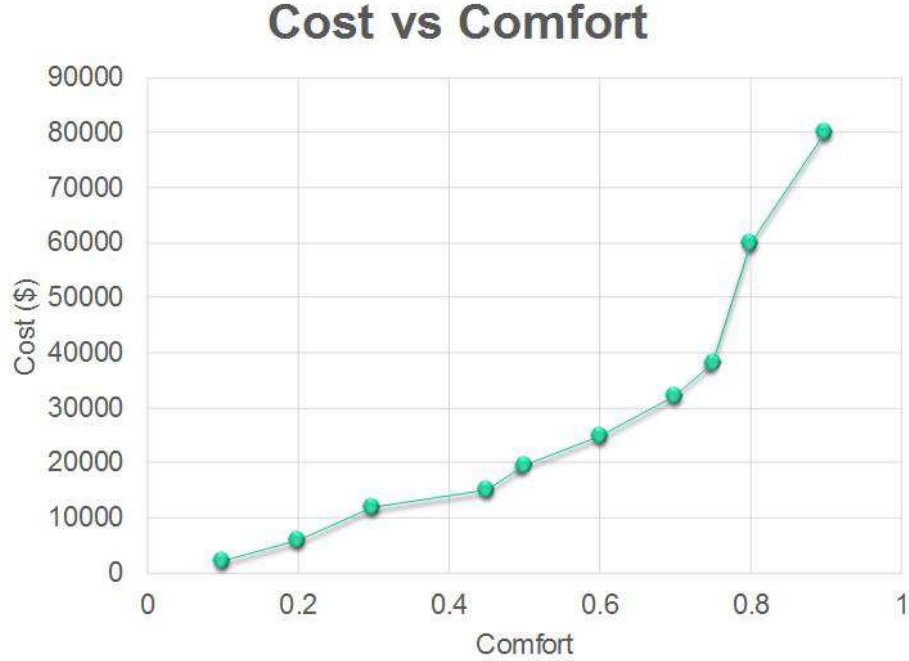


Figure 5: Pareto set of solutions for a theoretical cost-comfort car purchasing scenario.

An example of this is given in Figure 5, where a set of solutions for a car purchasing cost vs. comfort multi-objective problem are shown. As there is usually more than a single solution in the Pareto optimal set, population-based algorithms are a good fit for multi-objective optimization problems as they are well suited for providing a group of solutions, rather than just one.

2.3.1 Multi-Objective Optimization Algorithms

There are two fundamental ways to solve a multi-objective problem: find a Pareto optimal set of solutions and select one, either by a human decision maker or some pre-defined mechanism, or create a scalarization of the problem to transform it into a single-objective problem [60]. Scalarizing requires an in-depth understanding of the problem as well as a guarantee that the importance of each facet of the original multi-objective problem is static. If it is appropriate to scalarize a problem then any other single-objective optimization method can be applied to the scalarized problem. This methodology will lead to a single “optimal” solution for the problem. In the case where the requirements

of a solution are more dynamic, the decision maker may want the ability to select from a group of ideal solutions based on the current situation; this lends itself to the use of true multi-objective optimization algorithms as a set of solutions is produced.

Population-based multi-objective optimization algorithms provide a set of solutions to the decision maker based on the best solutions found during the algorithm's execution. The best solutions are determined using *Pareto dominance*. A solution is said to dominate another if it is superior in at least one way, and not inferior in any [14]. This concept of dominance is also used as the crux of many algorithms as well. **Non-dominated Sorting Genetic Algorithm II** (NSGA-II) [14], a commonly used multi-objective genetic algorithm, is based around moving forward between iterations with only the least dominated solutions. **Strength Pareto Evolutionary Algorithm 2** (SPEA2) [20], another commonly used algorithm, uses an internal strength function based on fitness, solution distribution within the solution space, and solution dominance to determine which solutions to carry forward. Both of these algorithms present their set of non-dominated solutions at the final iteration allowing the decision maker to select based on their current situation. NSGA-II and SPEA2 are both multi-objective genetic algorithms, which are a type of evolutionary multi-objective optimization (EMOO) algorithms. Another population-based algorithm, is particle swarm optimization (PSO) [61]. PSO works by having a population of solutions, referred to as a swarm, called particles. The particles move around the solution space based on the guidance received from other particles and their own track record. When a particle is deemed to have a better solution the other particles will move towards them. In the multi-objective variants, the Pareto dominance of a particle is taken into account when the particles move. A major difference between multi-objective PSO and multi-objective genetic algorithms is that the former do not rely on the genetic operators (i.e., crossover and mutation) that are the crux of the latter. With regards to input types; multi-objective PSO does not work with permutation-based solutions, or other discrete-type inputs. There are exceptions to this such as in [62] wherein discrete functions are mapped to continuous functions and then demapped after and [63] which redefines operators based on sets.

2.3.2 Genetic Algorithms

Genetic algorithms are a type of evolutionary algorithm that can be used to solve a wide range of optimization problems without making assumptions on the form of the underlying function (i.e., differentiability, continuity, etc.). They take their inspiration from the sharing of genetic information between generations of sexually-reproducing species. The core concept is that given two parent solutions, we can create two offspring solutions that represent a mixture of the parents. The nature of GAs makes them a form of stochastic algorithms, i.e., are never guaranteed the same output, even from the same starting point, since random decisions, such as whether to mutate a particular chromosome or not, will be made throughout the algorithm's execution. This effect is especially pronounced due to the divide between the genotype (the internal representation of the solution) and the phenotype (the external representation of the solution). When offspring solutions are generated, the genotypes are mixed and we evaluate the solution based on the new phenotype.

The inherent problem is that mixing two good genotypes does not guarantee a good phenotype in an offspring solution. This is the same in nature, breeding two healthy animals does not guarantee a healthy offspring; however, this may also be due to a mutation, a randomly occurring change in the genotype. GAs incorporate mutation into their design; they can be helpful instead of harmful. From an optimization perspective, they allow a random exploration of the solution space which can prevent the solution population from converging towards a local optimum. GAs can be broken down into four major building blocks: selection, crossover, mutation, and pruning, as detailed in Figure 6.

The selection mechanism of a GA determines which individuals in the solution population will become parent solutions. There are many mechanisms such as **random selection**, **roulette wheel selection** [64], **tournament selection** [65], and **elitist selection** [66]. Random selection works by arbitrarily picking any two individuals from the population for reproduction. Roulette selection assigns each solution a probability based on its fitness value; better solutions get a higher chance of being selected. Tournament

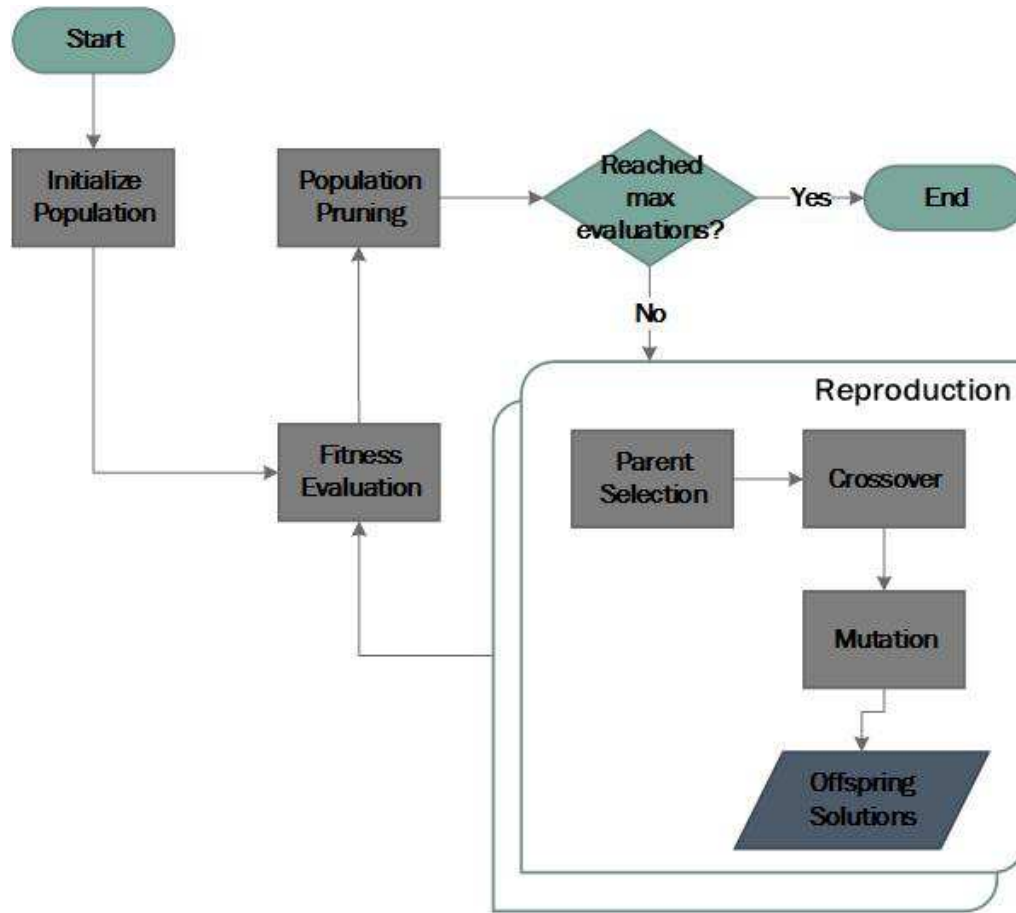


Figure 6: The flow of a typical genetic algorithm

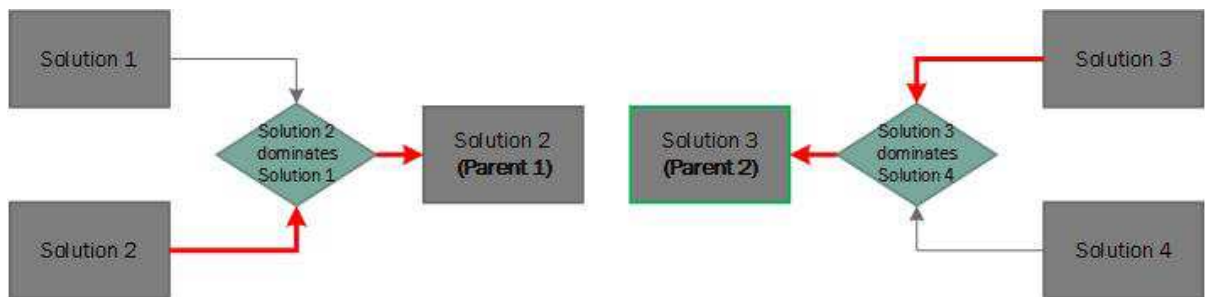


Figure 7: An example of binary tournament selection for determining the parents of potential offspring solutions

selection randomly chooses a pre-determined number of solutions from the population and puts them in tournament style brackets, as illustrated in Figure 7. This is done twice in order to select two parent solutions for reproduction. Finally, elitist selections will only use the top percentage of the population and employ one of the other selection

methods using the pool of solutions determined by the elitist method.

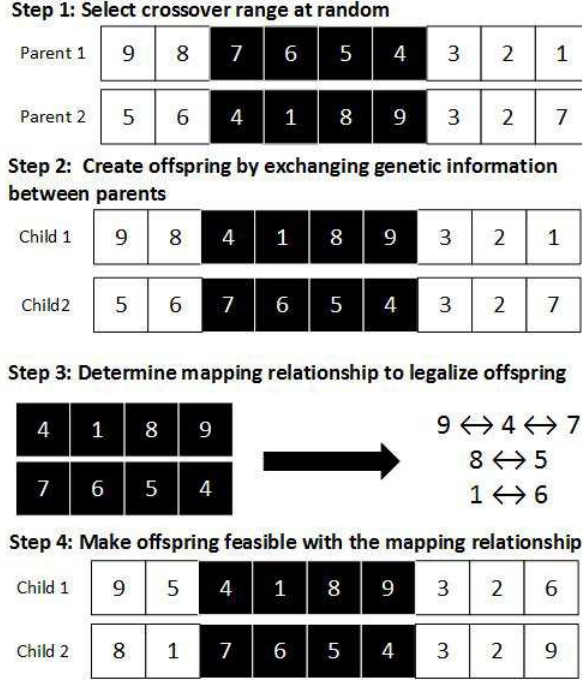


Figure 8: An example of Partially Mapped Crossover.

Once two solutions have been selected as parents, they must undergo crossover in order to create two offspring solutions. There are many different types of crossover operators that apply to many types of solution representations. GAs can be employed on almost any optimization problem so long as they have the proper representation. This can include permutations, bit-string, integers and real numbers, strings, and trees. Even programs can be created this way with a method called **genetic programming** [67]. Combinatorics problems often rely on permutation solution representation due to their nature as each possible permutation represents a different solution. With regards to crossover for permutations there are a few possible options such as: **partially mapped crossover** (PMX) [64], **cycle crossover** (CX) [64], and **order crossover** (OX) [64]. PMX works by randomly selecting two crossover points and exchanging the sections within the points. Any duplicate entries are replaced using a mapping from the parents; an example of PMX is shown in Figure 8. CX uses the idea of cycles found between the two parent solutions to create offspring solutions. To create a cycle we look at an index of parent 1 and note the contents, and look at the same index of parent 2. We look at the

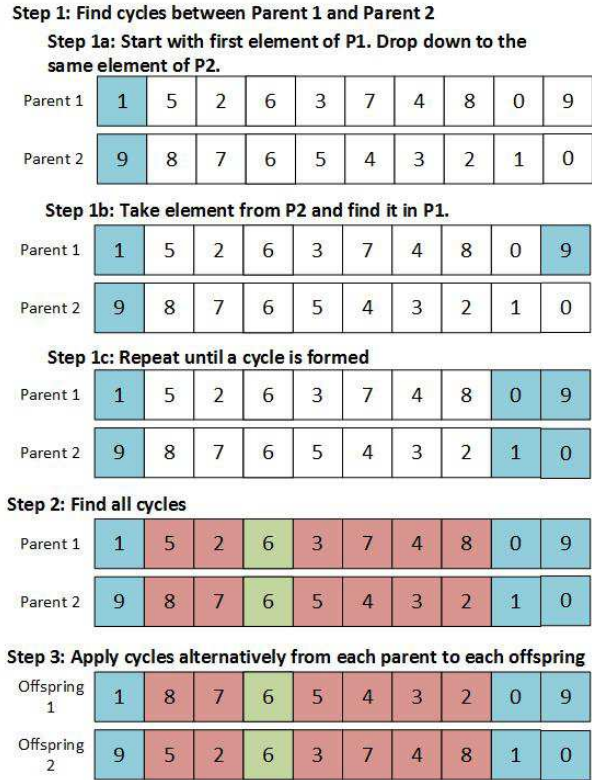


Figure 9: An example of Cycle Crossover.

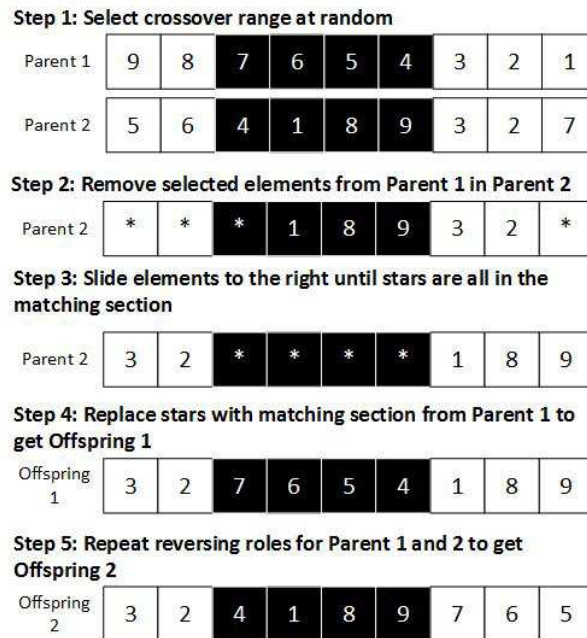


Figure 10: An example of Order Crossover.

element at that index and find it in parent 1. We then look at the element in the same index at parent 2. This is repeated until a cycle of elements is created. CX is illustrated for clarity in Figure 9. OX works similarly to PMX, but does not use a mapping for replacing the duplicate elements that are in the subsection; instead, OX will remove any duplicate elements and “slide” the remainder of the elements to the right, looping around the permutation if need be. The subsection is then placed into the shifted permutation, thus creating an offspring solution. This is illustrated in Figure 10.

The creation of offspring solutions via crossover does not mark the end of the reproduction process. Just like in nature, there can be an element of surprise in reproduction via mutation. In the case of a GA, this represents an opportunity for some randomized exploration of the solution space, similar to **simulated annealing** algorithms [68]. Generally, mutation only has a small chance of occurring, preventing the GA from getting too far off track. Permutation-based algorithms provide a unique challenge in that all elements of the permutation must be present in a solution, preventing the modification of elements without consequences. There are two basic forms of mutation that are viable for permutation-based solutions: **swap** and **insertion**. A swap mutation will switch the elements at two randomly chosen indexes of a permutation. An insertion mutation will take a random element of the permutation and randomly insert it elsewhere in the permutation. **Local search** algorithms can also be seen as a form of mutation. These algorithms examine the offspring solution and modify it based on its phenotype. Some examples of this are **k-opt** algorithms [69], and **tabu search** [70], while not explicitly a local search method, is used as one here.

The final step of a GA generation is pruning the population. Pruning is done to keep the population at a reasonable size. GAs are very flexible in terms of their parameters, but for simplicity we will assume that a GA will create a number of offspring solutions equal to its population; therefore, after reproduction there will be twice the initial number of solutions in the population. There are a few different options for pruning the population back down to the original size in preparation for the next generation. Moving forward with only the new offspring solutions is one option; another is to merge the populations and take only the best half of solutions. Randomly selecting the new

population is possible as well. Another approach is immediately after reproduction, take the offspring or parent solutions depending on which are better. The various components of a GA each provide many different options and provide many possibilities for different types of GA implementations, even when applying them to the same problem.

2.4 Risk-Aware Systems

Risk-aware systems provide a way for systems to autonomously determine the risk in a given scenario [71]. In order for this to occur the system must have some way of quantifying the risk of a given situation. This requires that the input to a risk-aware system must also be quantifiable so that the underlying mathematical model of the system can evaluate it and output a meaningful risk value to the user. These systems must also be assessed using some performance metrics so that their effectiveness within their application domain can be determined.

The mathematical model behind a risk-aware system is its backbone. It determines the type and format in input and output data, as well as how the risk is determined. Some models are quantifiable only at some steps of evaluation whereas others are quantifiable at each step. There are systems that use hidden Markov-chains, a stochastic state-based model, as the crux of their model [72]. Markov-chains represent a series of states and use probabilities for moving between various states. The model proposed by Tan et al. [72] is a general one, and does not indicate how probability values are determined. An alternative to this would be fuzzy sets, which allow for partial set membership, as seen in [17]. Keramati and Akbari propose a model which employs a statistical method which determines its probabilities based on the Common Vulnerability Scoring System² (CVSS) [73]. CVSS uses a set of metrics that can be divided into three groups: base, temporal, and environmental metrics. The combination of these metrics determines the current risk and assigns a score between 0 and 10. Each of these approaches is designed to be appropriate to its domain, from E-government network security, to wireless sensor networks, to computer networks [71].

Within the realm of risk management, metrics can be difficult to calculate. Since there

²<https://www.first.org/cvss>

is no overarching risk management framework, there are no unified metrics that can be used to fairly compare all risk management systems [71]. Some metrics are defined in such a way to be able to capture a specific task. This can be seen in [74] where the relevant task is the management of risk in global software engineering projects. Ryan and Ryan outline another approach that uses efficiency and effectiveness metrics that capture how a risk management system can extend the life of critical assets [75]. Another approach is given in [76] where the authors offer a system that decomposes threat scenarios into results and events. By breaking a threat scenario down into small quantitative pieces it becomes simpler to measure each component of a given threat scenario.

Risk-aware systems pose a unique challenge in their design and application as they must be tailored to their domain of application. As each system is unique in purpose, it is not possible to provide a homogeneous set of metrics across all risk-aware implementations.

A Multi-Objective Optimization Approach to Reliable Robot-Assisted Sensor Relocation

3.1 Preamble

This chapter contains the paper “*A Multi-Objective Optimization Approach to Reliable Robot-Assisted Sensor Relocation*” [1]. This paper introduces the *Reliable Robot-Assisted Sensor Relocation* (RRASR) problem as an extension of the RASR problem. This is done by incorporating additional problem objectives with the intent of creating a problem that has consideration for potential real-world application. This problem is mathematically formulated as a multi-objective optimization problem. Additionally, we empirically examine state-of-the-art evolutionary multi-objective optimization algorithms NSGA-II [14], NSGA-III [18], SPEA2 [20], and PESA-II [19] and apply them to this problem in

order to determine their efficacy on a large sample set of non-traditional benchmark instances. The experiments examine the effect of network size on the problem, and the performance of the algorithms on the problem with varying degrees of inflicted damage on the network. Some traditional multi-objective optimization metrics, such as: hypervolume [77], spacing [77], generational distance [77], inverted generational distance [78], maximum pareto front error [77], and runtime are use to evaluate algorithm performance. Our findings indicate that PESA-II is the best overall performing algorithm for the RRASR problem and provide insights into the performance of all algorithms across the two experiments.



RightsLink®

[Home](#)
[Create Account](#)
[Help](#)


Title: A multi-objective optimization approach to reliable Robot-Assisted Sensor Relocation

Conference Proceedings: 2015 IEEE Congress on Evolutionary Computation (CEC)

Author: Benjamin Desjardins; Rafael Falcon; Rami Abielmona; Emil Petriu

Publisher: IEEE

Date: 25-28 May 2015

Copyright © 2015, IEEE

[LOGIN](#)

If you're a [copyright.com](#) user, you can login to RightsLink using your copyright.com credentials. Already a [RightsLink user](#) or want to [learn more?](#)

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

[BACK](#)
[CLOSE WINDOW](#)

Copyright © 2016 [Copyright Clearance Center, Inc.](#) All Rights Reserved. [Privacy statement](#). [Terms and Conditions](#). Comments? We would like to hear from you. E-mail us at customercare@copyright.com

A Multi-Objective Optimization Approach to Reliable Robot-Assisted Sensor Relocation

Benjamin Desjardins, Rafael Falcon, Rami Abielmona and Emil Petriu

Abstract

Mobile robots can provide a wide range of value-added services to wireless sensor networks during their operational lifetime. One of them has to do with the replacement of damaged sensors with other functional, passive ones. This scenario has been recently studied as the “*Robot-Assisted Sensor Relocation*” (RASR) problem. RASR solutions traditionally focus only on one aspect of the problem: minimizing the total distance traveled by the robot. The existing centralized solutions do not take into account the reliability of the passive nodes that are selected as substitutes for the damaged nodes, for instance, their current battery level. With this in mind, we propose a *multi-objective optimization* (MOO) formulation of the problem, named Reliable Robot-Assisted Sensor Relocation (RRASR), where we consider two more objectives in addition to the trajectory length. These objectives result from the fact that a given passive sensor selected to replace a damaged sensor in the region may not be in perfect condition and that another passive sensor may be a better option. Due to the nature of MOO problems, we must present a diverse set of solutions that exhibit a trade-off among the different decision objectives to a network manager so they may take appropriate action. We evaluate the performance of four state-of-the-art evolutionary MOO algorithms with sensor networks of varying sizes and inflicted damage levels. To the best of our knowledge, this is the first time RASR is approached from an MOO angle.

Index Terms

evolutionary multi-objective optimization; genetic algorithms; wireless sensor and robot networks; robotics; wireless sensor networks; sensor relocation

I. INTRODUCTION

Wireless Sensor Networks (WSNs) consist of a group of wireless sensors that are deployed in a region of interest (ROI) [1]. A wireless sensor is defined as a device equipped with some form of wireless communication, one or more pieces of monitoring hardware, and a battery [1]. These sensors are deployed to gather low-level information in a ROI for high-level applications in domains such as medicine, defence, and infrastructure [1]. As such, these sensors must be deployed in such a way that there are no internal sensing holes so as to provide complete and uninterrupted surveillance.

Wireless Sensor and Robot Networks (WSRNs) [2] are an extension of WSNs wherein a robot, or group of robots, perform some sensing and/or actuation task related to the WSN. In these cases, while the sensors may be limited in their capabilities, the robots are flexible, generally mobile with fewer or no resource limitations and are designed to assist, maintain, and/or optimize the sensor network. According to [3], WSRNs can be divided into two categories: robot-assisted WSNs, and robot-dependent WSNs. In the former case, the robot(s) are not integral to the system, but their functionality will enhance its overall performance. In the latter case, the robot(s) are required for the system to remain functional.

Robot-Assisted Sensor Relocation (RASR), also referred to in the literature as “*carrier-based coverage repair*” [4], is a specific WSRN-related problem in which a robot is responsible for maintaining the network. In this problem, the robot(s), on standby at a common base station, are responsible for relocating sensors in the network in order to maximize network performance. RASR assumes that there are an abundance of sensors deployed in the ROI in excess of what is necessary to achieve maximum coverage. These excess sensors are not required to actively collect information and will henceforth be referred to as passive sensors. These passive sensors are used by the robot(s) to replace damaged sensors. The robot(s) will leave the base station to reconfigure sensors when there has been enough change in the network; generally, a change refers to a currently active sensor being damaged or its battery running out. Determining the required amount of change that triggers a robot deployment is outside the scope of this study.

We propose an approach to the RASR problem in which a single mobile robot, able only to carry a limited number of sensors, is located at the base station. The goal of the RASR problem is to find an optimal sensor replacement trajectory (in terms of distance traveled by the mobile robot), and as such can be expressed as an optimization problem, the *one-commodity traveling salesman problem with selective pickup and delivery*, as seen in [4]. The issue with the approach as seen in [4] is that it does not consider the quality of the passive sensors picked up or their placement. This poses a problem because, in the

B. Desjardins and E. Petriu are with the School of Electrical Engineering & Computer Science, University of Ottawa, Ottawa ON, K1N 6N5 Canada {bdesj038, petriu}@uottawa.ca

R. Falcon and R. Abielmona are with the Research & Engineering Division, Larus Technologies Corporation, Ottawa ON, K1P 5V5 Canada {rafael.falcon, rami.abielmona}@larus.com

TABLE I
RELATED RASR LITERATURE

Reference	Architecture	Number of Robots	Solution	Assumptions
[4]	Centralized	One	Ant Colony System	Robot has no battery constraints. Robot can only carry a limited number of sensors.
[5]	Centralized	One	Ant Colony System + Constrained Neighbourhood Mutation (ACS+CNM)	Robot has no battery constraints. Robot can only carry a limited number of sensors.
[6]	Centralized	Many	Firefly Algorithm & Harmony Search	Robot has no battery constraints. Robot can only carry a limited number of sensors.
[7]	Localized	One	Proactive & reactive replacement algorithms	Robot may only carry one sensor. Robot has battery life. Robot wanders when not recharging.
[8]	Localized	Many	Localized Ant-based Sensor Relocation Algorithm with Greedy Walk (LASR-G)	Robot may only carry one sensor. Maximizes coverage in any situation be it node failure or poor deployment
[9]	Centralized	One	Dual-objective genetic algorithm	No relocation of nodes, only replacing batteries. Batteries deplete over the course of replacement
[10]	Localized	One or Many	Market-based Sensor Relocation (MSR)	Robot may only carry one sensor.

worst case, the robot could place a passive sensor with low battery such that it would fail immediately after placement. Due to this issue we approach RASR from a *multi-objective optimization* (MOO) perspective which, to the best of our knowledge, is being done for the first time. We not only consider the trajectory length but also the lifetime of the resulting replacement trajectory, and a metric we propose called the *trajectory robustness*, as two additional decision objectives. These objectives are based on the previously mentioned issues: passive sensors have varying battery levels and therefore the choice of a particular passive sensor makes a difference in the resulting sensor network. We refer to the new problem as *Reliable Robot-Assisted Sensor Relocation* (RRASR) and use *evolutionary multi-objective optimization* (EMOO) algorithms to solve it.

In this paper we make the following contributions: (1) we review the most relevant works in the literature from a twofold standpoint: sensor relocation by mobile robots and the novel optimization problem that emerges as a result of it; (2) we extend and formalize the RASR problem to accommodate more realistic scenarios that directly impact the reliability of the proposed solution; (3) we cast the emerging RRASR problem into the multi-objective optimization realm by introducing additional decision objectives like trajectory robustness and trajectory lifetime; (4) we solve this problem using EMOO algorithms and design their building blocks; (5) we discuss the results of a multi-angled empirical performance analysis of four well-known EMOO algorithms that deal with sensor networks of varying sizes and inflicted damage levels.

The rest of the manuscript is structured as follows. Section II briefly reviews relevant works. Section III formalizes the RRASR problem while Section IV elaborates on the algorithmic building blocks for the EMOO schemes under consideration. Section V is concerned with the empirical evaluation of the methodology being put forth. Finally, Section VI concludes the paper.

II. RELATED WORK

This Section briefly reviews relevant studies concerning robot-assisted sensor relocation and its underlying optimization backbone.

A. Robot-Assisted Sensor Relocation

Various forms of RASR have been studied using different methods, as can be seen in Table I. Some of these formulations, such as [4], [5], [6], and [9] envision a centralized system where the robots operate out of a central base station. Only [6], [8], and [10] provide solutions with multiple robots to repair the WSN. Magklara et al [7] provide a restriction on the robot wherein it has battery life. All of the formulations place a restriction on how many sensors may be carried by the robot(s), be it one sensor or greater. [10] is unique among the localized solutions in that the sensors themselves help play a role in helping the robot to identify the location of the sensing hole.

The formulation provided by [9] differs from the other substantially in that it only replaces the batteries of dead sensors causing sensing holes. It bears mentioning as it has the solution closest to ours in the RASR space. It uses a genetic algorithm to solve a pseudo MOO problem where one objective takes priority over another. However, the first objective exists only to make sure that solutions include all sensing holes, rather than add depth to the problem.

We solve this problem by adding more depth to the scenario to more realistically model the problem. In doing so we differ from the other RASR solutions by incorporating an MOO approach that results from the consideration given to the additional depth of the scenario.

We extend the problem solved in [4] by adding more decision objectives to the selection of the optimal sensor replacement trajectories. In doing so, we differ from existing RASR solutions in that we incorporate a MOO approach to more realistically model RASR.

B. Related Optimization Problems

TABLE II
RELATED OPTIMIZATION LITERATURE

Reference	Optimization Problem	Optimization Algorithm	Objective	Assumptions
[11]	Selective Pickup and Delivery Problem (SPDP)	Genetic Algorithm (GA)	Minimize route length	No maximum vehicle load.
[12]	Multi-vehicle SPDP	GA with Path Relinking	Minimize sum of route lengths	All vehicles are used in each instance.
[13]-1	SPDP	GA with Adaptive Mutation	Minimize route length	No maximum vehicle load.
[13]-2	SPDP	GA with Local Search and Adaptive Mutation	Minimize route length	No maximum vehicle load.
[14]	Single Vehicle Routing Problem with Deliveries and Selective Pick-ups (SVRPDSP)	GA with Variable Neighbourhood Search	Minimize route length	Multiple visits to delivery location is valid. Allows pickup and delivery from same location.
[15]	SVRPDSP	Evolutionary Algorithm with Variable Neighbourhood Descent	Minimize route length	Multiple visits to delivery location is valid. Allows pickup and delivery from same location.
[16]	SPDP	GA with Local Search	Minimize route length	No maximum vehicle load.
[17]	Multiple Vehicle Routing Problem with Delivery and Selective Pick-ups (MVRPDSP)	Hybrid Metaheuristic	Minimize sum of route lengths	Allows pickup and delivery from same location.

A generalization of RASR, in which pickup and delivery locations can store more than one item, has also been studied as an optimization problem under different names: *selective pickup and delivery problem* (SPDP) [11] and the *Single Vehicle Routing Problem with Deliveries and Selective Pickups* (SVRPDSP) [14]. They refer to versions of the *Traveling Salesman Problem* (TSP) and the *vehicle routing problem* (VRP) respectively in which all delivery demands must be satisfied, but not all pickup locations must be visited. An overview of these problems can be seen in Table II-B. Since TSP and VRP are known to be NP-hard, we can say that RASR is also NP-hard.

There have been studies conducted using GAs to find optimal solutions to a given problem instance. Liao and Ting take a standard GA approach to this problem in [11]. Notably they use a repair function in order to deal with infeasible solutions. They expand on their approach in [16] by using a memetic algorithm in order to improve their solutions. It is worth mentioning that improvements to the local search will improve the entire algorithm. Liao and Ting also look to adaptive mutation to improve solutions from both a standard GA approach and a memetic algorithm approach in [13]. A study by Bruck et al uses a GA with variable neighbourhood search in order to find a solution to the problem [14]. This is expanded in [17] where solutions for the multiple vehicle version of the problem are sought. Huang and Ting also proposed a solution to the multiple vehicle version of the problem in [12]. Bruck and dos Santos expand again on [14] in [15] where they use a different gene representation for their GA as well as using data mining techniques in order to improve their crossover and mutation operators.

The major difference in our approach is the addition of multiple objectives. This forces us to take a different perspective of the problem as none of the approaches mentioned in the literature handle this problem. Due to the nature of MOO problems we must look at a different way of solving the problem than those already presented. The genetic representations and operators used in [11], [16], [13], [12], [14], and [15] can still be considered relevant as it would be possible to implement them in an MOO algorithm.

III. RRASR: A MULTI-OBJECTIVE RASR FORMULATION

Our problem can be formulated as a combinatorial optimization problem. We represent our sensor network as a complete undirected graph $G = (V, E)$ with vertex set $V = \{v_0, \dots, v_n\}$ and edge set $E = \{e_{ij} = (v_i, v_j) | v_i, v_j \in V, v_i \neq v_j\}$ in which each edge e_{ij} has a cost $d_{ij} > 0$ representing the Euclidean distance between v_i and v_j .

Each vertex represents either a passive sensor or a sensing hole and has an associated unitary demand q_i (1 for passive sensors and -1 for sensing holes). The base station is denoted as v_0 with $q_0 = 0$. Therefore we can say that any other vertex either belongs to the set of passive sensors $S = \{v_i | v_i \in V, q_i = 1\}$ or the set of sensing holes $H = \{v_i | v_i \in V, q_i = -1\}$. This implies that $V = \{S \cup H \cup v_0\}$.

Each sensor has an associated power level denoted by

$$p_i = \begin{cases} 0 & \text{if } v_i \in H \\ -1 & \text{if } i = 0 \\ \sim U(0, 100) & \text{if } v_i \in S \end{cases}$$

There is a unique commodity (sensors) to be transported by the robot from one place to another. The robot can carry at most Q_{max} sensors and leaves the base station with an initial cargo Q_0 , $0 \leq Q_0 \leq Q_{max}$. The robot will always drop its most recently picked up sensor when it reaches a sensing hole due to simulated design constraints on the robotic arm and mobile platform.

We also consider the set of all active nodes $A = \{a_0, \dots, a_m, v_0\}$ that form the WSN deployed in the ROI. This consideration is important because it allows us to determine the importance of a node in the WSN with regards to maintaining network connectivity. v_0 is specially included in A to account for those sensing holes only known to the base station.

All passive and damaged sensors $v_i \in V, 1 < i < n$ have a communication radius of R . We use the edge set $E' = \{e'_{ij} = (a_i \in A, v_j \in H)\}$ to find the active node degree g_j of each sensing hole $v_j \in H$ such that $g_j = |\{a_i \in A : d'_{ij} \leq R\}|$ and d'_{ij} stands for the Euclidean distance between a_i and v_j where $a_i \in A, v_j \in H$.

We want to find a feasible sensor relocation trajectory, $\varphi = (v_{\gamma_1}, v_{\gamma_2}, \dots, v_{\gamma_{|\varphi|}})$ where $\gamma([1; |V| - 1]) \rightarrow \{\gamma_i\}$ is an injective function and $v \in V \setminus \{v_0\}$, that starts and ends at the base station. As stated by Falcon et al. in [4], a sensor relocation trajectory is said to be feasible if it has no repeated nodes (other than the base station as its first and last element), repairs all sensor holes (i.e., drops a passive sensor at the location of each sensing hole), and never violates the robot's capacity constraint. The quality of φ will be evaluated according to the following objectives:

Minimize Trajectory Length

$$\sum_{e_{ij} \in \varphi} d_{ij} \quad (1)$$

Maximize Trajectory Robustness

$$\sum_{v_i \in \varphi \cap S, v_j \in \varphi \cap H} \frac{p_i}{g_j} \delta_{ij} \quad (2)$$

Maximize Trajectory Lifetime

$$\min\{p_i\}, v_i \in \varphi \cap S \quad (3)$$

The trajectory robustness is representative of how likely a sensor hole would be to cause a disconnect in the WSN. It is expressed as a ratio between the battery power of the passive sensor being placed in the sensing hole and the active degree of said sensing hole. The trajectory lifetime shows the worst passive sensor power level in a sensor relocation trajectory. The function δ_{ij} is evaluated as follows:

$$\delta_{ij} = \begin{cases} 1 & \text{if } v_i \in \varphi \cap S \text{ will be dropped off at the} \\ & \text{location of } v_j \in \varphi \cap H \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

IV. EMOO ALGORITHMS FOR RRASR

In this Section we unveil several common building blocks of the EMOO algorithms that will be applied to solve the RRASR problem.

A. Solution Encoding

We use the modified version of a permutation-based representation proposed in [11]. For a problem instance with n nodes we will use a permutation of $n - 1$ integers excluding the base station node, v_0 , from the permutation). Each solution will take the form of $\{v_1, \dots, v_n\}$ to represent the visiting order of the nodes. Not all nodes are required to be visited, therefore unvisited nodes will be denoted by a negative integer. For example, the solution $(-4, 2, 5, -3, 1, 6)$ indicates that the robot will visit the nodes 2, 5, 1 and 6 in that order. Since RRASR solutions must be quickly computed and then enacted on the WSN, we adopted this permutation-based representation as it ensures higher computational efficiency at the expense of greater memory capacity.

B. Objective Functions

We use an MOO approach in which fitness functions (1), (2), and (3) are evaluated and have no precedence over each other. As a result, each solution is assessed in terms of three different (and conflictive) objectives.

C. Population Initialization

The population is initialized by creating three heuristically determined individuals, one for each of the objective functions and then randomly initializing the remainder of the population. For trajectory length (1) we use a nearest neighbour heuristic in which we always take the shortest available and valid edge. For trajectory robustness (2) we always pick the sensor with the highest battery level and deliver it to the sensing hole with the smallest degree regardless of location. Finally, the heuristic for trajectory lifetime (3) is to select the available passive sensor with highest battery for placement at each sensing hole, regardless of location.

The rest of the population is randomly initialized by incrementally and arbitrarily adding the elements of the permutation. Their sign (positive or negative) is determined with equal probability. To prevent infeasible solutions from entering the population, all randomly generated solutions undergo the repair operation, as seen in section IV-E, before entering the population.

D. Evolutionary Operators

1) *Selection Operator*: We use binary tournament selection in which two individuals are randomly chosen and the best solution from the pair will be selected. Since this is an MOO problem the non-domination rank [18] of a given solution, rather than a single fitness function, will be used for comparison. In the case that two competing solutions share the same rank either solution will be chosen with equal probability.

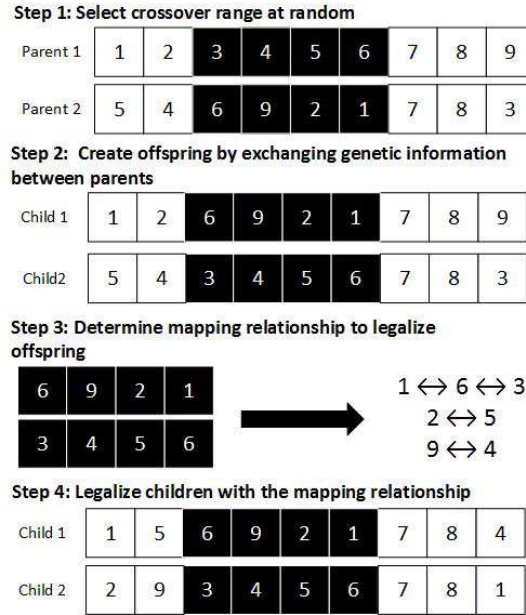


Fig. 1. Example of partially-mapped crossover

2) *Crossover Operator*: Partially-mapped crossover (PMX) [19] is used as the crossover operator. This type of crossover builds offspring by selecting a subsequence of a solution from one parent and preserving the order and position of as many points as possible from the other parent, as shown in Fig. 1.

3) *Mutation Operator*: Two mutation operators are used: *swap* and *insert*. Each offspring may undergo a mutation. The *swap* operator works by randomly selecting two indexes of the permutation and switching their values. The *insert* operator works by removing an item from the permutation and inserting it into a random new location in the permutation.

Generic swap and insert mutations have no effect on the solution vector if only negative elements are affected. In order to make the mutations more effective, whenever an element of the permutation is affected by a mutation it also has its sign flipped. These mutations were noted to have more of an effect on solutions as they guaranteed changes in the solution vector.

E. Infeasibility Handling

We look at the constraints of the problem and repair a given individual if it violates them. In the case of the sensor relocation trajectory not having the right number and type of nodes, we use the following methods to ensure that a solution will be viable.

- In the case $H \subseteq \varphi$ is not true, we add the remaining elements of H to φ . We do this by making all elements in H in the encoded solution positive if they are not already.

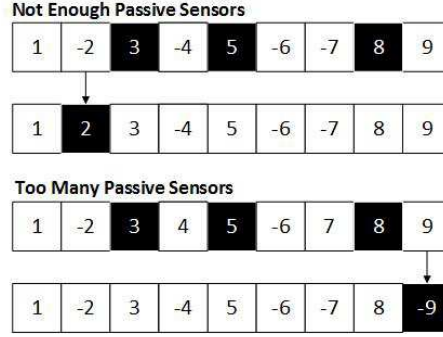


Fig. 2. In the first example there are not enough passive sensors in the relocation trajectory. We add passive sensors from right to left until we have enough. In the second example there are too many passive sensors in the relocation trajectory. We remove passive sensors from left to right.

- If $|\varphi| > 2|H| - Q_0$, remove $v_i \in S \cap \varphi$ from right to left until $|\varphi| = 2|H| - Q_0$. By going from right to left we prevent passive sensors from being collected after the last sensing hole is visited. We do this by iterating through the permutation from n to 1 and flipping the sign of those v_i from positive to negative until the condition is satisfied.
- If $|\varphi| < 2|H| - Q_0$, add $v_i \in S, v_i \notin \varphi$ from left to right until $|\varphi| = 2|H| - Q_0$. By going from left to right we prevent the removal of passive sensor collections that are being used as part of the solution. We do this by iterating through the permutation from 1 to n and flipping the sign of those v_i from negative to positive until the condition is satisfied.

This portion of the repair function can be seen in Fig. 2.

The other constraint that must be considered for repair is: at all points in φ , $0 \leq Q \leq Q_{max}$ must hold. We must consider the two infeasible cases:

- If $Q > Q_{max}$ at some point in φ , move the offending $v_i \in \varphi \cap S$ after the nearest right $v_i \in \varphi \cap H$ in φ . To do this we insert the vertex in which the issue appears after the rightmost element of H that appears in the permutation.
- If $Q < 0$ at some point in φ , move the offending $v_i \in \varphi \cap H$ after the nearest right $v_i \in \varphi \cap S$ in φ . To do this we insert the vertex in which the issue appears after the rightmost element of S that appears in the permutation.

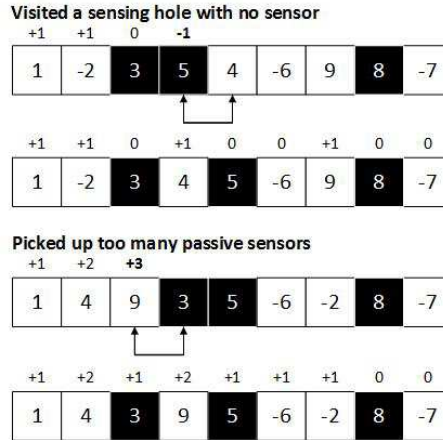


Fig. 3. Examples of sensor relocation trajectory ordering repair for $Q_{max} = 2$. In the first example the robot attempts to place a sensor when it is not carrying one. We swap the offending sensing hole with the next passive sensor. In the second example the robot attempts to collect a passive sensor when it is already carrying a full load. We swap the offending passive sensor with the next sensing hole.

This portion of the repair function can be seen in Fig. 3.

After each of the above repair steps the infeasibility item in question will have been resolved.

F. Stop Criteria

All algorithms will terminate after a set number of objective function evaluations has occurred in order to ensure a fair empirical comparison baseline. In a real-world scenario, a time-based stop criterion would be more appropriate as RRASR is generally time-constrained.

V. EXPERIMENTAL RESULTS

This Section elaborates on the empirical evaluation of the proposed MOO methodology for the RRASR problem.

A. Experimental Setup

1) *Synthetic Scenario Generation*: The scenarios used for testing the proposed approach have been generated with the ability to control certain parameters such as the number of nodes N , the number of passive sensors $NP = |S|$ and the ratio of sensing holes to passive sensors R_{HS} . In order to simulate a RRASR scenario, we generate the scenario by first mapping a grid of active sensors that would represent the sensor network in the ROI. We then take a subset of those active sensors and turn them into sensing holes. Finally, we randomly distribute the remainder of the nodes as passive sensors using the MAX_DPA algorithm adapted from [20]. The resulting set of passive sensors and sensing holes is the scenario used for experimentation.

We generated 98 scenarios for the following parameters: N from 10 to 1,000 in increasing steps for a total of 49 scenarios; R_{HS} from 1% - 49% for a total of 49 scenarios.

2) *Algorithm and Parameter Configurations*: Four state-of-the-art EMOO algorithms were selected for the empirical analysis: NSGA-II [18], SPEA2 [21], PESA-II [22], and NSGA-III [23]. In order to generate some performance metrics, we first created reference Pareto sets for each scenario as follows: each of the four aforesaid EMOO algorithms was run 50 times on the scenario and the resulting 200 Pareto sets were merged. The non-dominated front of this set became the reference Pareto optimal set for that RRASR scenario.

The parametric configuration common to all EMOO algorithms is given in Table III.

TABLE III
DEFAULT EMOO ALGORITHM PARAMETERS

Parameter	Value
Population Size	200
Nr. of Function Evaluations	100000
Probability of Crossover	0.75
Probability of Mutation	0.25

We conducted the experiments using their MOEA Framework [24] implementations. For PESA-II and SPEA2, we ported its jMetal [25] implementation to MOEA Framework. The crossover and mutation probabilities are the MOEA Framework default values and we decided to use those for our experiments.

3) *Performance Metrics*: Table IV lists the six MOO performance metrics used in this study.

The reported metric values are the averages, over all synthetic RRASR scenarios, of the means of 30 independent runs of an algorithm upon a scenario.

TABLE IV
ALGORITHM PERFORMANCE METRICS

Name	Ref.	Goal	Description
Hypervolume	[26]	Max	Represents the volume of the objective space dominated by solutions in the approximation set.
Max Pareto Front Error	[26]	Min	Represents the maximum distance from solutions in an approximation set to the nearest solution in the reference set.
Spacing	[26]	Max	Represents the spread of the Pareto approximation set.
Generational Distance	[26]	Min	Represents average distance from solutions in the approximation set to the nearest solution in the reference set.
Inverted Generational Distance	[27]	Min	Represents the average distance from solutions in the reference set to the nearest solution in the approximation set.
Elapsed Time	N/A	Min	The runtime (in seconds) of the algorithm.

4) *Statistical Validation*: To confirm whether statistically significant differences in the performances of the algorithms tested exist or not, we rely on nonparametric tests as suggested in [28]. In all cases a significance level of 5% is used.

As we do not have a control algorithm, we first establish or reject the existence of statistically significant differences in the whole group using the Friedman $N \times N$ procedure. In the case that the null hypothesis is rejected (i.e., there exists some statistically significant differences in the group), then a set of pairwise comparisons is carried out using the following post-hoc procedures: Nemenyi, Holm, Schaffer, and Bergmann [28]. We will employ Bergmann as the default post-hoc procedure given its robustness and reliable adjustment of the p -value. Only when Bergmann does not reject a hypothesis will we resort to the other post-hoc methods.

The algorithm performance data used for the statistical validation procedure was collected at 25, 50, 75, and 100 percent of the number of function evaluations across each of the scenarios.

B. Experiment 1: Scalability Analysis

In this experiment we test the effect of WSN size on our RRASR solutions. We set the percentage of sensing holes to 15 for this experiment.

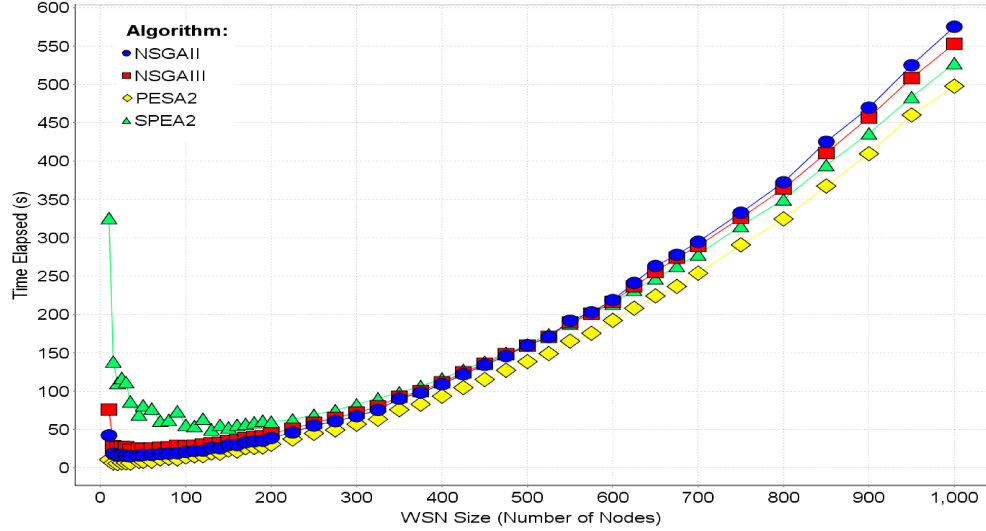


Fig. 4. Final execution results for time elapsed, in seconds, across all generated scenarios for the scalability analysis experiment

Fig. 4 displays the running time rendered by each algorithm as a function of the number of WSN nodes. SPEA2 is notable in that it performs very poorly at small WSNs but better as the WSN size increases. This is likely due to the distance function SPEA2 uses in order to try and keep diverse solutions. When the number of possible solutions is small, SPEA2's archive becomes full of similar or duplicate solutions which causes an uncharacteristically large amount of comparisons to be done for each solution. This could be the cause of the disparity between SPEA2 and the other algorithms in terms of runtime.

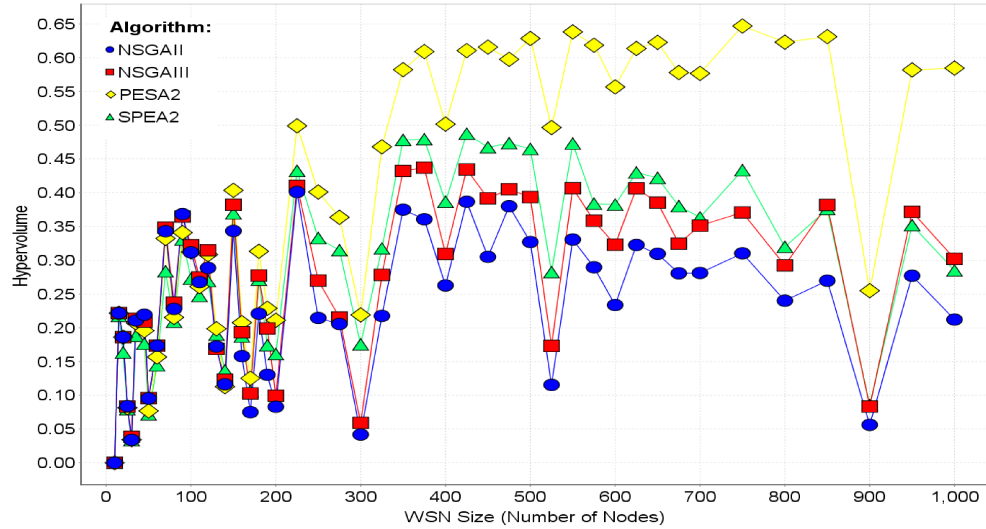


Fig. 5. Final execution results for hypervolume, across all generated scenarios for the scalability analysis experiment

Fig. 5 shows the Hypervolume over the same domain as Fig. 4. We can see that PESA-II finds consistently superior results as compared to the other algorithms. This is likely due to the size of the hypergrid that PESA-II uses for solution grouping. Only non-dominated solutions are put into this hypergrid. Fitness is then determined by a *squeeze factor* which is equivalent to the number of solutions in the hyperbox. Solutions with low squeeze factors are then selected for reproduction. The size

of the hypergrid can determine the diversity in solutions sets as well as execution time. In the future it would be interested to further investigate the effect of the size of the hypergrid on algorithm performance.

NSGA-II likely has such poor performance due to the fact that it does not discriminate, beyond selection, when choosing candidates for reproduction. Any individual in the population is eligible to be chosen whereas the other algorithms are more strict about which individuals are eligible for reproduction.

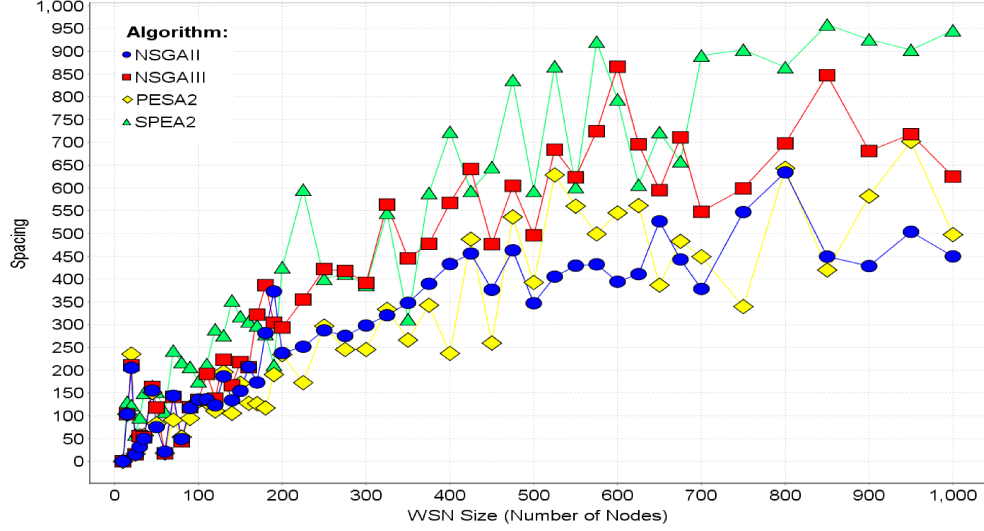


Fig. 6. Final execution results for spacing, across all generated scenarios for the scalability analysis experiment

Fig. 6 shows the metric where PESA-II does not outperform the other algorithms: spacing. We can see that SPEA2 consistently outperforms the other algorithms in this regard. The distance function that SPEA2 uses for breaking ties when putting individuals into its archive set maximizes the distance between solutions, which Fig. 6 clearly shows the benefit of. The major difference in the implementation of NSGA-II and SPEA2 is the set of individuals they select parents from. NSGA-II can select parents from its entire population whereas SPEA2 may only select parents from its archive set. Since the archive set of SPEA2 is designed to contain the best, most diverse solutions it makes sense that SPEA2 performs better than the others. PESA-II does not have any spacing or crowding distance calculation which might be the reason why it performs poorly in this regard.

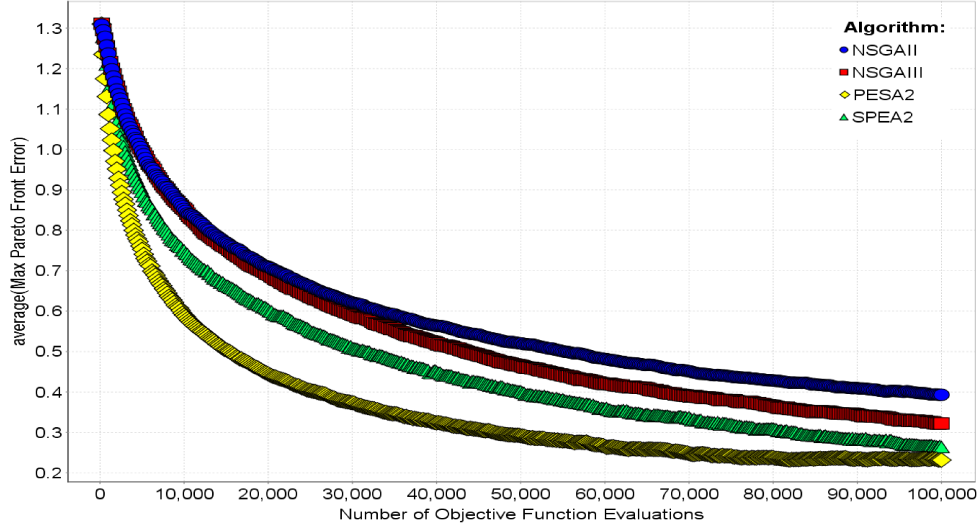


Fig. 7. Average, across all generated scenarios, results for the maximum Pareto front error for the scalability analysis experiment

Fig. 7 portrays the mean maximum Pareto front Error achieved by each algorithm throughout its execution over all scenarios.. What is interesting here is how NSGA-II and NSGA-III perform very similarly for the first quarter of the run and then diverge as time goes on. NSGA-II and NSGA-III both use the spacing metric but NSGA-III is much more selective in which individuals are allowed to reproduce. This is likely the cause of the divergence: as more good solutions become available NSGA-III is more likely to give them reproductive preference than NSGA-II is.

We can also notice the convergence of SPEA2 and PESA-II near the end of the execution. PESA-II progresses very quickly towards good solutions but likely gets stuck in some local optima in the solution space, the lack of spacing shown by PESA-II is evidence for this. SPEA2 consistently uses its best, most diverse solutions for reproduction. The trend shown by SPEA2 is much smoother than the one shown by PESA-II perhaps indicating superior performance for longer runs than we used in our experiments.

TABLE V
EXPERIMENT 1 - FRIEDMAN $N \times N$ TEST RESULTS

Metric	Algorithm Rank				p -value
	NSGA-II	NSGA-III	SPEA2	PESA-II	
Hypervolume	3.3597	2.574	2.6378	1.4286	1.0567E-10
Spacing	3.273	2.0893	1.3852	3.2526	1.5823E-10
Max Pareto Front Error	3.6224	2.8469	2.0689	1.4617	1.6400E-10
Generational Distance	2.8393	2.3699	3.4158	1.375	1.4274E-10
Inverted Generational Distance	2.3801	3.1658	2.6582	1.7959	9.6787E-11
Runtime	2.4949	3.0357	3.4694	1	1.9837E-10

Table V gives an overview of the results found for the Friedman procedure on each performance metric. The small p -values indicate the existence of statistically significant differences among the algorithm group for each metric under consideration. The only null hypotheses that were not rejected by the Bergmann procedure, as well as any other post-hoc procedure, were NSGA-III vs. SPEA2 for hypervolume and NSGA-II vs. PESA-II for spacing. As neither of these instances include the best ranked for that metric, we can trust our results in showing which algorithm performs the best for each metric.

C. Experiment 2: Inflicted Damage Analysis

In this experiment we study the performance of the EMOO methods in a WSN composed of 200 static nodes as an increasing number of them become sensing holes, e.g., in case of a malicious attack.

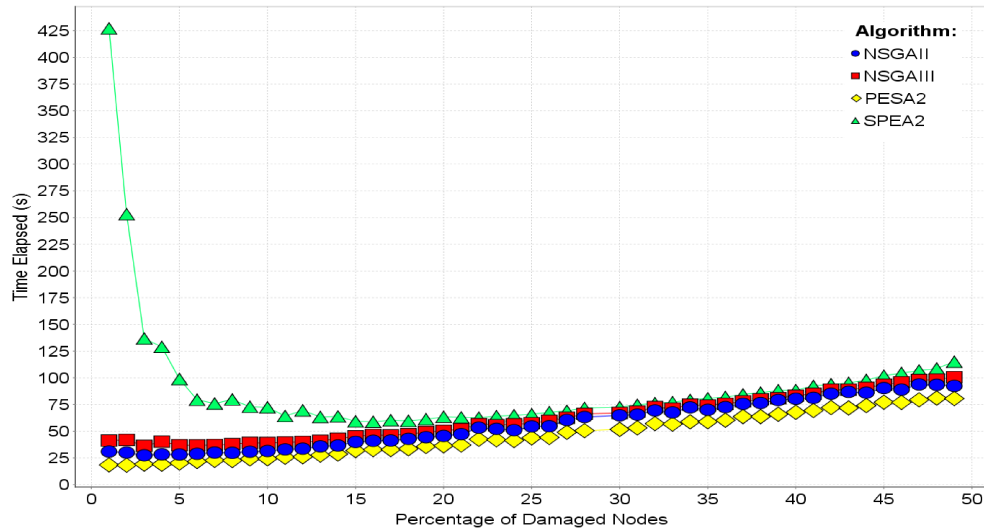


Fig. 8. Final execution results for time elapsed, in seconds, across all generated scenarios for the inflicted damage analysis experiment

Fig. 8 unveils a smooth linear dependency between the percentage of damaged nodes and the time needed to compute RRASR solutions for all algorithms except SPEA2, which suffers from poor diversification in small search spaces as pointed

out before. The more sensing holes are present in the WSN, the larger the ensuing sensor relocation trajectories will be and hence the longer it will take to explore the bigger solution spaces induced.

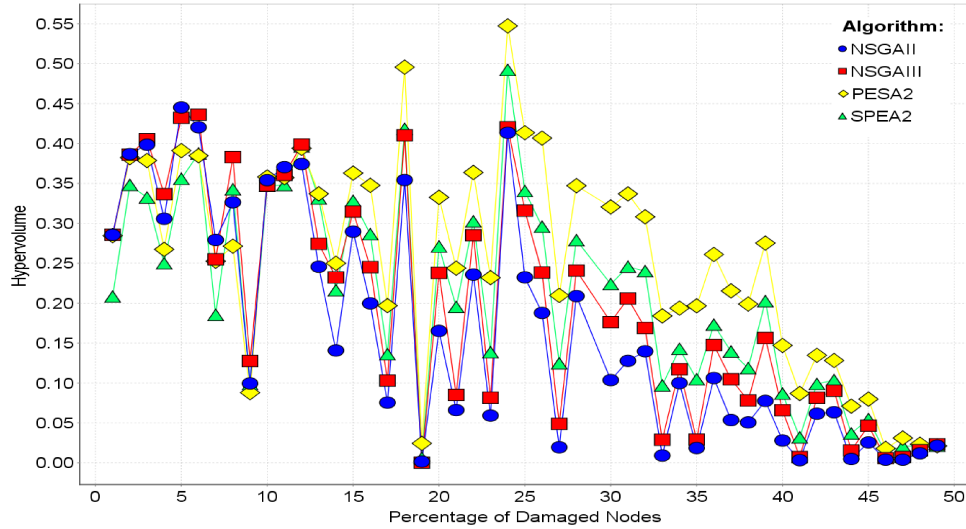


Fig. 9. Final execution results for hypervolume, across all generated scenarios for the inflicted damage analysis experiment

Fig. 9 reveals that for WSNs with up to 15% of their nodes being damaged, nearly all algorithms manage to obtain similar hypervolume values. The contrast among them grows starker as the inflicted damage level rises, with PESA-II discovering the best non-dominated fronts given that [give an insight here]. As the number of sensing holes continues to mount up, we observe a decline in the obtained hypervolume values, i.e., the EMOO schemes deal with more complex search spaces that cannot be well approximated with the limited budget of number of function evaluations assigned to each.

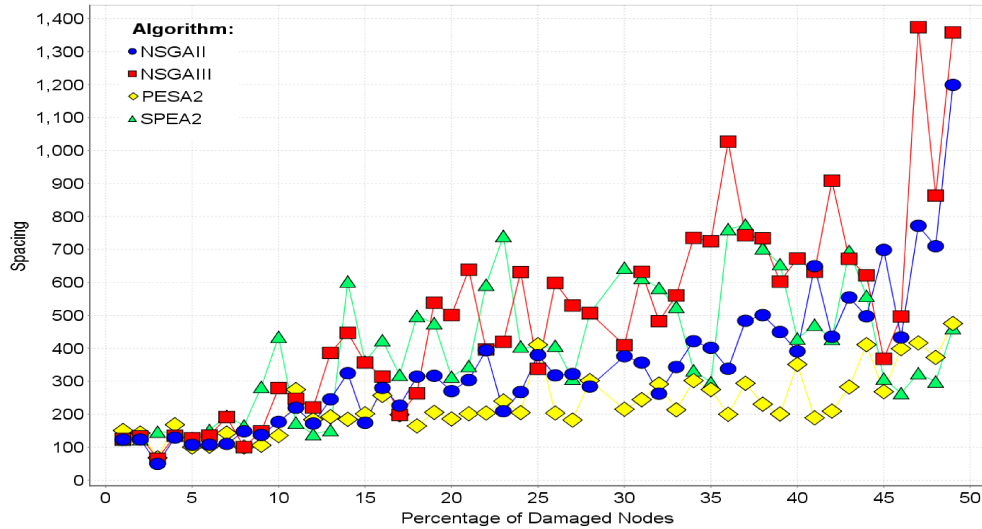


Fig. 10. Final execution results for spacing, across all generated scenarios for the inflicted damage analysis experiment

As the solution space, and the number of possible solutions, increases the spacing of the solutions found by the algorithms does as well. The upward trends in the spacing values depicted in Fig. 10 reveal this fact, as the ratio of sensing holes increases. The greater the solution space, the higher the number of solutions that will be found causing the increase in spacing values provided by each algorithm's diversification mechanisms.

Fig. 11 shows trends very similar to those from Fig. 7. This indicates that the algorithm performance in these two experiments is more related to the actual algorithm's design features than the problem space itself.

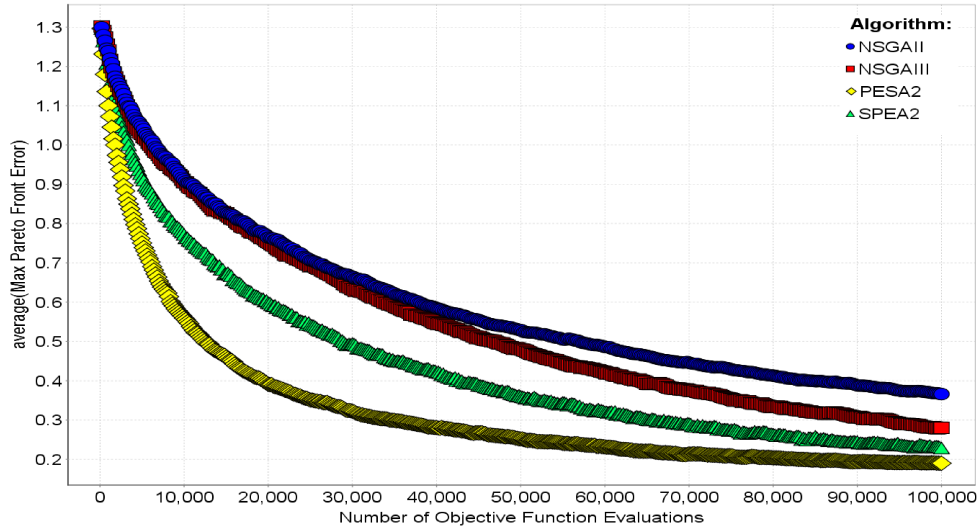


Fig. 11. Average results, across all generated scenarios, for the maximum Pareto front error in the inflicted damage analysis experiment

TABLE VI
EXPERIMENT 2 - FRIEDMAN $N \times N$ TEST RESULTS

Metric	Algorithm Rank				p -value
	NSGA-II	NSGA-III	SPEA2	PESA-II	
Hypervolume	3.3854	2.875	2.3594	1.3802	1.2717E-10
Spacing	2.9635	1.6302	1.8438	3.5625	1.5111E-10
Max Pareto Front Error	3.7448	2.9583	2.0052	1.2917	2.0198E-10
Generational Distance	3.0625	2.7865	2.9479	1.2031	1.3301E-10
Inverted Generational Distance	2.9219	3.3698	2.3646	1.3438	1.1218E-10
Runtime	2	3	4	1	2.2089E-10

Table VI offers a very similar view of the statistical validation as that in Table V, with PESA-II being the best algorithm across all metrics except spacing. For these results the Bergmann procedure or any other post-hoc procedures do not reject the following null hypotheses: NSGA-III vs. SPEA2 for spacing, and NSGA-II vs. NSGA-III, NSGA-III vs. SPEA2, and NSGA-II vs. SPEA2 for generational distance. The algorithm rankings for generational distance indicate that PESA-II once again outperforms the others, but the Bergmann procedure tells us that we can't compare the other algorithms with any certainty. We can again see that PESA-II performed better than any of the other algorithms overall.

VI. CONCLUSIONS

We modeled a new version of the RASR problem, which we called RRASR, in order to generate more reliable sensor relocation trajectories for the mobile robot. RRASR was formulated as an MOO problem to be tackled via EMOO algorithms, whose genetic operators were also given in this work. Four state-of-the-art EMOO methods were tested with WSNs of different sizes and inflicted damage levels. Our results show that PESA-II performs the best on these problems with the caveat that it shows the least diversity in solutions. As future work, we would like to incorporate additional decision objectives to make RRASR more relevant to real-world WSRN deployments. We would also like to further investigate and analyze the EMOO methods tested with regards to RRASR.

REFERENCES

- [1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.
- [2] W. Shi and J.-P. Corriveau, "A comprehensive review of sensor relocation," in *In Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing (GREENCOM-CPSCOM '10)*, pp. 780–785, 2010.
- [3] R. Falcon, *Towards Fault Reactiveness in Wireless Sensor Networks with Mobile Carrier Robots*. PhD thesis, University of Ottawa, Ottawa, Ontario, Canada, 2012.
- [4] 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 *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, (Barcelona, Spain), pp. 4326–4333, 2010.

- [5] M. Lian-Ming and D. Xi-Li, "A novel ant colony system for solving the one-commodity traveling salesman problem with selective pickup and delivery," in *Natural Computation (ICNC), 2012 Eighth International Conference on*, pp. 1096–1101, IEEE, 2012.
- [6] R. Falcon, X. Li, A. Nayak, and I. Stojmenovic, "A Harmony-Seeking Firefly Swarm to the Periodic Replacement of Damaged Sensors by a Team of Mobile Robots," in *2012 IEEE International Conference on Communications (ICC)*, (Ottawa, Canada), pp. 6436–6440, June 2012.
- [7] K. Magklara, D. Zorbas, and T. Razafindralambo, "Node discovery and replacement using mobile robot," in *Ad Hoc Networks*, pp. 59–71, Springer, 2013.
- [8] Y. Wang, A. Barnawi, R. F. De Mello, and I. Stojmenovic, "Localized ant colony of robots for redeployment in wireless sensor networks," *Multi-Valued Logic & Soft Computing*, vol. 23, pp. 35–51, 2014.
- [9] Y. Miao and W. Yu-Ping, "Coverage repair strategies for wireless sensor networks using mobile actor based on evolutionary computing," *Bulletin of Electrical Engineering and Informatics*, vol. 3, no. 3, pp. 213–222, 2014.
- [10] H. Li, A. Barnawi, I. Stojmenovic, and C. Wang, "Market-based sensor relocation by robot team in wireless sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 22, pp. 259–280, 2014.
- [11] X.-L. Liao and C.-K. Ting, "An evolutionary approach for the selective pickup and delivery problem," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pp. 1–8, IEEE, 2010.
- [12] Y.-H. Huang and C.-K. Ting, "Genetic algorithm with path relinking for the multi-vehicle selective pickup and delivery problem," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pp. 1818–1825, June 2011.
- [13] X.-L. Liao and C.-K. Ting, "Evolutionary algorithms using adaptive mutation for the selective pickup and delivery problem," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pp. 1–8, IEEE, 2012.
- [14] B. P. Bruck, A. G. dos Santos, and J. E. C. Arroyo, "Hybrid metaheuristic for the single vehicle routing problem with deliveries and selective pickups," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pp. 1–8, IEEE, 2012.
- [15] B. P. Bruck, A. Santos, and J. Arroyo, "An evolutionary algorithm and a variable neighborhood descent algorithm for the single vehicle problem with deliveries and selective pickups," *Proceedings of the 2012 CLAIO/SBPO (Rio de Janeiro, Brazil)*, 2012.
- [16] C.-K. Ting and X.-L. Liao, "The selective pickup and delivery problem: Formulation and a memetic algorithm," *International Journal of Production Economics*, vol. 141, no. 1, pp. 199–211, 2013.
- [17] B. P. Bruck and A. dos Santos, "Hybrid approach for the multiple vehicle routing problem with deliveries and selective pickups," in *Hybrid Intelligent Systems (HIS), 2012 12th International Conference on*, pp. 265–270, Dec 2012.
- [18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, Apr. 2002.
- [19] D. E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," in *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, vol. 154, Lawrence Erlbaum, Hillsdale, NJ, 1985.
- [20] F. A. Onat, I. Stojmenovic, and H. Yanikomeroglu, "Generating random graphs for the simulation of wireless ad hoc, actuator, sensor, and internet networks," *Pervasive and Mobile Computing*, vol. 4, no. 5, pp. 597–615, 2008.
- [21] E. Zitzler, M. Laumanns, L. Thiele, E. Zitzler, E. Zitzler, L. Thiele, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," 2001.
- [22] D. W. Corne, N. R. Jerram, J. D. Knowles, M. J. Oates, *et al.*, "PESA-II: Region-based selection in evolutionary multiobjective optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2001)*, pp. 283–290, Morgan Kaufmann Publishers, 2001.
- [23] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *Evolutionary Computation, IEEE Transactions on*, vol. 18, pp. 577–601, Aug 2014.
- [24] D. Hadka *et al.*, "MOEA framework." <http://www.moeaframework.org>.
- [25] A. Nebro and J. Durillo, "jmetal framework." <http://jmetal.sourceforge.net>.
- [26] C. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007.
- [27] M. R. Sierra and C. A. C. Coello, "Improving pso-based multi-objective optimization using crowding, mutation and-dominance," in *Evolutionary multi-criterion optimization*, pp. 505–519, Springer, 2005.
- [28] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

Planning Robust Sensor Relocation Trajectories for a Mobile Robot with Evolutionary Multi-Objective Optimization

4.1 Preamble

This chapter contains the book chapter “*Planning Robust Sensor Relocation Trajectories for a Mobile Robot with Evolutionary Multi-Objective Optimization*” [2]. This chapter expands on the work contained in the previous chapter by updating the trajectory robustness objective function as well as integrating a *risk-management framework* (RMF) for proactive sensor replacement. The utilization of the RMF allows the detection of active sensors that have exceeded a permissible risk threshold and are therefore flagged as

damaged nodes, hence becoming the target of a new sensor relocation round by the mobile robot. Additionally, two more evolutionary multi-objective optimization algorithms, AGE-I [21] and AGE-II [22], are added to the evaluation group. Parametric tuning is conducted in order to evaluate each algorithm using a set of input parameters suited to it. We conduct three experiments in order to test algorithm performance: the effect of network size on the problem, and the performance of the algorithms on the problem with varying degrees of inflicted damage on the network, and the physical density of the passive sensors within the network. We evaluate the performance of all algorithms using the same set of metrics used in [1]. We find that PESA-II and NSGA-III are the best overall performing algorithms. AGE-I and AGE-II claim to perform better than all of the other algorithms included here on a set of standard benchmark problems. We show here that on an applied problem this is not the case.

Title of the Book or Conference Name: Computational Intelligence in Wireless Sensor Networks: Recent Advances and Future Challenges

Volume Editor(s):

Title of the Contribution: Planning Robust Sensor Relocation Trajectories for a Mobile Robot with Evolutionary Multi-Objective Optimization

Author(s) Name(s): Benjamin Desjardins, Rafael Falcon, Rami Abielmona, Emil Petriu

Corresponding Author's Name, Address, Affiliation and Email: Benjamin Desjardins

When Author is more than one person the expression "Author" as used in this agreement will apply collectively unless otherwise indicated.

§ 1 Rights Granted

Author hereby grants and assigns to Springer International Publishing AG, Cham (hereinafter called Springer) the exclusive, sole, permanent, world-wide, transferable, sub-licensable and unlimited right to reproduce, publish, distribute, transmit, make available or otherwise communicate to the public, translate, publicly perform, archive, store, lease or lend and sell the Contribution or parts thereof individually or together with other works in any language, in all revisions and versions (including soft cover, book club and collected editions, anthologies, advance printing, reprints or print to order, microfilm editions, audiograms and videograms), in all forms and media of expression including in electronic form (including offline and online use, push or pull technologies, use in databases and networks for display, print and storing on any and all stationary or portable end-user devices, e.g. text readers, audio, video or interactive devices, and for use in multimedia or interactive versions as well as for the display or transmission of the Contribution or parts thereof in data networks or search engines), in whole, in part or in abridged form, in each case as now known or developed in the future, including the right to grant further time-limited or permanent rights. For the purposes of use in electronic forms, Springer may adjust the Contribution to the respective form of use and include links or otherwise combine it with other works. For the avoidance of doubt, Springer has the right to permit others to use individual illustrations and may use the Contribution for advertising purposes.

The copyright of the Contribution will be held in the name of Springer. Springer may take, either in its own name or in that of copyright holder, any necessary steps to protect these rights against infringement by third parties. It will have the copyright notice inserted into all editions of the Contribution according to the provisions of the Universal Copyright Convention (UCC) and dutifully take care of all formalities in this connection in the name of the copyright holder.

§ 2 Regulations for Authors under Special Copyright Law

The parties acknowledge that there may be no basis for claim of copyright in the United States to a Contribution prepared by an officer or employee of the United States government as part of that person's official duties. If the Contribution was performed under a United States government contract, but Author is not a United States government employee, Springer grants the United States government royalty-free permission to reproduce all or part of the Contribution and to authorize others to do so for United States government purposes.

If the Contribution was prepared or published by or under the direction or control of Her Majesty (i.e., the constitutional monarch of the Commonwealth realm) or any Crown government department, the copyright in the Contribution shall, subject to any agreement with Author, belong to Her Majesty.

If the Contribution was created by an employee of the European Union or the European Atomic Energy Community (EU/Euratom) in the performance of their duties, the regulation 31/EEC, 11/EAEC (Staff Regulations) applies, and copyright in the Contribution shall, subject to the Publication Framework Agreement (EC Plug), belong to the European Union or the European Atomic Energy Community.

If Author is an officer or employee of the United States government, of the Crown, or of EU/Euratom, reference will be made to this status on the signature page.

§ 3 Rights Retained by Author

Author retains, in addition to uses permitted by law, the right to communicate the content of the Contribution to other scientists, to share the Contribution with them in manuscript form, to perform or present the Contribution or to use the content for non-commercial internal and educational purposes, provided the Springer publication is mentioned as the

28.01.2014
14:07

original source of publication in any printed or electronic materials. Author retains the right to republish the Contribution in any collection consisting solely of Author's own works without charge subject to ensuring that the publication by Springer is properly credited and that the relevant copyright notice is repeated verbatim.

Author may self-archive an author-created version of his/her Contribution on his/her own website and/or the repository of Author's department or faculty. Author may also deposit this version on his/her funder's or funder's designated repository at the funder's request or as a result of a legal obligation. He/she may not use the publisher's PDF version, which is posted on SpringerLink and other Springer websites, for the purpose of self-archiving or deposit. Furthermore, Author may only post his/her own version, provided acknowledgment is given to the original source of publication and a link is inserted to the published article on Springer's website. The link must be provided by inserting the DOI number of the article in the following sentence: "The final publication is available at Springer via [http://dx.doi.org/\[insert DOI\]](http://dx.doi.org/[insert DOI])". The DOI (Digital Object Identifier) can be found at the bottom of the first page of the published paper.

Prior versions of the Contribution published on non-commercial pre-print servers like ArXiv/CoRR and HAL can remain on these servers and/or can be updated with Author's accepted version. The final published version (in pdf or html/xml format) cannot be used for this purpose. Acknowledgment needs to be given to the final publication and a link must be inserted to the published Contribution on Springer's website, by inserting the DOI number of the article in the following sentence: "The final publication is available at Springer via [http://dx.doi.org/\[insert DOI\]](http://dx.doi.org/[insert DOI])".

Author retains the right to use his/her Contribution for his/her further scientific career by including the final published paper in his/her dissertation or doctoral thesis provided acknowledgment is given to the original source of publication. Author also retains the right to use, without having to pay a fee and without having to inform the publisher, parts of the Contribution (e.g. illustrations) for inclusion in future work, and to publish a substantially revised version (at least 30% new content) elsewhere, provided that the original Springer Contribution is properly cited.

§4 Warranties

Author warrants that the Contribution is original except for such excerpts from copyrighted works (including illustrations, tables, animations and text quotations) as may be included with the permission of the copyright holder thereof, in which case(s) Author is required to obtain written permission to the extent necessary and to indicate the precise sources of the excerpts in the manuscript. Author is also requested to store the signed permission forms and to make them available to Springer if required.

Author warrants that he/she is entitled to grant the rights in accordance with Clause 1 "Rights Granted", that he/she has not assigned such rights to third parties, that the Contribution has not heretofore been published in whole or in part, that the Contribution contains no libelous statements and does not infringe on any copyright, trademark, patent, statutory right or proprietary right of others, including rights obtained through licenses; and that Author will indemnify Springer against any costs, expenses or damages for which Springer may become liable as a result of any breach of this warranty.

§5 Delivery of the Work and Publication

Author agrees to deliver to the responsible Volume Editor (for conferences, usually one of the Program Chairs), on a date to be agreed upon, the manuscript created according to the Springer Instructions for Authors. Springer will undertake the reproduction and distribution of the Contribution at its own expense and risk. After submission of the Consent to Publish form Signed by the Corresponding Author, changes of authorship, or in the order of the authors listed, will not be accepted by Springer.

§6 Author's Discount

Author is entitled to purchase for his/her personal use (directly from Springer) the Work or other books published by Springer at a discount of 33 1/3% off the list price as long as there is a contractual arrangement between Author and Springer and subject to applicable book price regulation. Resale of such copies or of free copies is not permitted.

§7 Governing Law and Jurisdiction

This agreement shall be governed by, and shall be construed in accordance with, the laws of Switzerland. The courts of Zug, Switzerland shall have the exclusive jurisdiction.

Corresponding Author signs for and accepts responsibility for releasing this material on behalf of any and all Co-authors.

Signature of Corresponding Author:

Date:

March 31, 2016

☐

☐

☐

I'm an employee of the US Government and transfer the rights to the extent transferable (Title 17 §105 U.S.C. applies)

I'm an employee of the Crown and copyright on the Contribution belongs to Her Majesty

I'm an employee of the EU or Euratom and copyright on the Contribution belongs to EU or Euratom

28.01.2014
14:07

Planning Robust Sensor Relocation Trajectories for a Mobile Robot with Evolutionary Multi-Objective Optimization

Benjamin Desjardins, Rafael Falcon, Rami Abielmona, Emil Petriu

Keywords: sensor relocation; wireless sensor and robot networks; multi-objective optimization; wireless sensor networks; genetic algorithms

Abstract. Wireless sensor networks provide a method for monitoring a region of interest. Incorporating a mobile robot within the sensor network allows various types of functionality to be added. One example of this is the replacement of risky and/or damaged sensors with other functional, passive ones. Using a specially designed *risk management framework* (RMF), we can proactively detect sensors that are at a high risk for failure and replace them before any network coverage is lost. The problem of optimizing the robot trajectory while picking up passive sensors and dropping them at the locations of the damaged sensors in the field has been studied as the “*Robot-Assisted Sensor Relocation*” (RASR) problem. One shortcoming of existing RASR solutions is that the chosen robot trajectory is the one with the shortest length; however, no regards as to the durability of the passive sensors in the relocation chain are taken into consideration. We propose a more robust manner to come up with these trajectories by taking into account the current energy levels of the participating passive sensors as well as the ideal locations for their deployment. We resort to *multi-objective optimization* (MOO) to handle the tradeoffs among the different decision objectives that are part of this new formulation, named here as “Reliable Robot-Assisted Sensor Relocation”. We outline the RRASR problem as well as the RMF used for detecting risky sensors in the wireless sensor network before the calculation of the sensor relocation trajectory takes place. We also evaluate the performance of six state-of-the-art *evolutionary multi-objective optimization* (EMOO) algorithms with sensor networks of varying sizes, inflicted damage levels, and passive sensor densities. The empirical results confirm the feasibility of utilizing EMOO approaches to suggest multiple sensor relocation trajectories to the network manager.

1 Introduction

A wireless sensor is a device that is equipped with monitoring hardware, a battery, and some form of wireless communication [1]. A group of these wireless sensors is referred to as a *Wireless Sensor Network* (WSN) and is deployed in a region of interest (ROI) [1]. WSNs gather low-level information about a ROI for high-level applications in various domains, including medicine, defense, and

agriculture [1]. As many of these domains are information-critical, the sensors must be deployed in such a way to provide complete sensing coverage over the ROI with no internal sensing holes.

WSNs can be functionally expanded with the addition of a mobile robot. This robot, or group of robots, perform some sensing or actuation task that is related to the WSN. This is referred to as a *Wireless Sensor and Robot Network* (WSRN) [2]. The robots add a flexibility not present in a WSN as they are generally mobile and have fewer or no resource restrictions and are designed to maintain, assist, or optimize the sensor network. Falcon explains in [2] that WSRNs can be divided into two categories: *robot-dependent WSNs*, and *robot-assisted WSNs*. In a robot-dependent WSN, the robot(s) are integral to the functionality of the network. However, robot-assisted WSNs do not require the functionality of robot(s), instead only relying on them to improve the performance of the network.

Robot-Assisted Sensor Relocation (RASR) is a specific WSRN-related problem wherein a robot is responsible for maintaining the coverage of the network. RASR is also referred to as “*carrier-based coverage repair*” [3] in the literature. This problem is defined by having a common base station where one or more robots are on standby. These robots are responsible for relocating and/or replacing sensors in order to maximize network performance. RASR assumes that there are additional sensors in the ROI not currently being used by the network to achieve maximum coverage. These extra sensors do not actively collect information and will be referred to as *passive sensors*. The robot(s) use these sensors to replace damaged sensors in the network, viz. by leaving the base station, picking up some passive sensors and dropping them at the locations of the damaged sensors. to reconfigure the network when there has been enough degradation of the coverage area.

This chapter builds upon the early work done by Desjardins et al. in [4] in defining *Reliable Robot-Assisted Sensor Relocation* (RRASR). The original work defines RRASR as a specific version of RASR wherein there is a single mobile robot that has a limited carrying capacity and our objective is to find an optimal sensor relocation trajectory. RRASR differs from RASR in that there are additional objectives of importance beyond trajectory length. These additional objectives are representative of the fact that when only considering the distance that the mobile robot travels, we ignore the quality of the sensors that are being placed. In the worst case scenario, a passive sensor with very little remaining battery life could be used to fill a sensing hole. This is a poor solution as it will require the robot to redeploy in a short amount of time to fix the same sensing hole. Due to these additional objectives, RRASR was formulated as a *multi-objective optimization* (MOO) problem and as such we use *evolutionary multi-objective optimization* (EMOO) algorithms to solve it.

In the original RRASR problem [4], robots are deployed when there are sensing holes already in the network. In this chapter, we present a method for identifying sensors that are likely to fail, and replacing them before a sensing hole occurs. This is essential for operations in critical systems where complete

sensing coverage of an ROI is always required. We rely on the *Risk Management Framework* (RMF), described in [5] to support the risk-aware analysis of which sensors become damaged at a certain point in time. Our tailored RMF formulation allows proactive replacement of sensors that are deemed at risk of failing or being faulty in order to try and minimize any lack of coverage in the ROI.

We expand on the original work in [4] with the following contributions: (1) review the literature on three fronts: sensor relocation by mobile robots, the optimization problem that stems from it, and the use of risk-driven schemes; (2) We propose a proactive replacement methodology using an RMF-based approach that monitors several key factors to ensure proper sensor functioning; (3) we modify one of the existing RRASR objectives proposed in [4] to more accurately reflect the importance of network connectivity when selecting what passive sensor to deploy at a certain spot; (4) we solve this problem using a set of state-of-the-art algorithms, including the newer AGE-I [6] and AGE-II [7] algorithms; (5) We use parametric tuning to ensure each algorithm provides the best possible results; (6) we discuss the results of our statistically validated empirical results, giving insight into the EMOO algorithms’ performance, in experiments dealing with various sizes of sensor networks, inflicted damage levels, and distribution density of passive sensors.

The remainder of the chapter is structured as follows: Section 2 briefly reviews relevant works. Section 3 details our proposed RMF formulation for determining the damaged sensors. Section 4 formalizes the RRASR problem and Section 5 elaborates on the algorithmic components for the EMOO techniques under consideration. Section 6 contains the empirical evaluation of the proposed methodology. Finally, Section 7 concludes the chapter.

2 Related Work

This section provides a brief overview of related studies regarding robot-assisted sensor relocation and its underlying optimization problem as well as the risk management framework.

2.1 Robot-Assisted Sensor Relocation

RASR has been studied in various forms using different methodologies, as summarized in Table 1. In addition to what method is used to solve the problem, there are two other categories that define an approach to RASR: the architecture, centralized or localized, and the number of robots being one or many. The formalizations provided in [3], [8], [9], and [13] use a centralized architecture, wherein a central server or base is responsible for determining the deployment and trajectory of the robot(s). In the centralized scenario, the robot(s) are stored in the base station until they are needed. Multiple-robot solutions are described in [9], [11], and [14] and use different methods for determining trajectories of the robots. All of the literature examined provided solutions where the robot has a limited carrying capacity. Uniquely, the solution provided by Li et al. in [14]

Table 1: Related RASR Literature

Ref.	Architecture	Number of Robots	Solution	Assumptions
[3]	Centralized	One	Ant Colony System	Robot has no battery constraints. Robot can only carry a limited number of sensors.
[8]	Centralized	One	Ant Colony System + Constrained Neighbourhood Mutation (ACS+CNM)	Robot has no battery constraints. Robot can only carry a limited number of sensors.
[9]	Centralized	Many	Firefly Algorithm & Harmony Search	Robot has no battery constraints. Robot can only carry a limited number of sensors.
[10]	Localized	One	Proactive & reactive replacement algorithms	Robot may only carry one sensor. Robot has battery life. Robot wanders when not recharging.
[11]	Localized	Many	Localized Ant-based Sensor Relocation Algorithm with Greedy Walk (LASR-G)	Robot may only carry one sensor. Maximizes coverage in any situation be it node failure or poor deployment
[12]	Localized	One or Many	Randomized Robot-assisted Relocation of Static Sensors (R3S2) and grid variant (G-R3S2)	Robot has no battery constraints Robot wanders when not actively replacing nodes
[13]	Centralized	One	Dual-objective genetic algorithm	No relocation of nodes, only replacing batteries. Batteries deplete over the course of replacement
[14]	Localized	One or Many	Market-based Sensor Relocation (MSR)	Robot may only carry one sensor.

uses a method where the sensors themselves play a role in helping the robot to identify sensing holes in the network. Fletcher et al. [12] use a strictly programmatic approach to replacement whereby one or many robots will wander a WSN and replace sensors as necessary. They provide two variants of their solution: grid-based and non-grid-based.

Within the RASR space, the solution closest to the one we propose is found in [13]. Coincidentally, it also differs the most from the others as it proposes a method where only the batteries of the sensors are replaced, rather than the sensors themselves. It uses a genetic algorithm (GA) to solve a two objective

MOO problem wherein one objective is favoured over the other. One of the objectives represents the portion of damaged sensors that are in the current solution. This acts as a driving factor for all damaged sensors to be included in the solutions, but does not add depth to the problem.

Our solution provides more depth to the problem by more realistically modeling it. We do this by incorporating an MOO approach that stems from the consideration given to other facets of the problem. We extend the problem solved in [3] by including additional decision objectives for determining the optimal sensor relocation trajectories. We set ourselves apart from traditional RASR solutions by incorporating additional decision objectives that provide a more realistic view of RASR.

2.2 Related Optimization Problems

Table 2: Related Optimization Literature

Ref.	Optimization Problem	Optimization Algorithm	Objective	Assumptions
[15]	Selective Pickup and Delivery Problem (SPDP)	Genetic Algorithm (GA)	Minimize route length	No maximum vehicle load.
[16]	Multi-vehicle SPDP	GA with Path Relinking	Minimize sum of route lengths	All vehicles are used in each instance.
[17]-1	SPDP	GA with Adaptive Mutation	Minimize route length	No maximum vehicle load.
[17]-2	SPDP	GA with Local Search and Adaptive Mutation	Minimize route length	No maximum vehicle load.
[18]	Single Vehicle Routing Problem with Deliveries and Selective Pickups (SVRPDSP)	GA with Variable Neighbourhood Search	Minimize route length	Multiple visits to delivery location is valid. Allows pickup and delivery from same location.
[19]	SVRPDSP	Evolutionary Algorithm with Variable Neighbourhood Descent	Minimize route length	Multiple visits to delivery location is valid. Allows pickup and delivery from same location.
[20]	SPDP	GA with Local Search	Minimize route length	No maximum vehicle load.
[21]	Multiple Vehicle Routing Problem with Deliveries and Selective Pickups (MVRPDSP)	Hybrid Meta-heuristic	Minimize sum of route lengths	Allows pickup and delivery from same location.

RASR can be generalized into versions of the *Travelling Salesman Problem* (TSP) and the *Vehicle Routing Problem* (VRP) optimization problems. The versions of these problems that most closely model RASR are *selective pickup and delivery problem* (SPDP) [15] and the *Single Vehicle Routing Problem with Deliveries and Selective Pickups* (SVRPDSP) [18]. The defining difference between TSP/VRP and SPDP/SVRPDSP is that all delivery requests must be satisfied, but not all pickup locations need to be visited. Table 2 gives an overview of some of the approaches to these problems. As a note, since TSP and VRP are known to be NP-hard, we can infer that RASR is NP-hard as well.

These problems have been approached in different ways, but of interest to us are approaches using evolutionary and genetic algorithms. Liao and Ting use a standard GA approach in [15]. An important feature of this work is that they incorporate a repair function in order to guarantee the feasibility of their solutions. This approach is expanded upon in [20] where they use a memetic algorithm in order to improve their results. In [17], Liao and Ting employ both a standard GA approach and a memetic approach with adaptive mutation to enhance solution quality. In [18], Bruck et al. use a GA with variable neighbourhood search. This is further developed in [21] where a multi-vehicle version of the problem is examined. The multi-vehicle problem is also examined by Huang and Ting in [16]. The approach in [18] is broadened on by Bruck and dos Santos in [19] where they explore a different gene representation as well as incorporate data mining techniques in order to improve their mutation and crossover operators.

Our approach diverges from the ones mentioned above due to the addition of multiple decision objectives. As the methods relating to this problem in the literature only address a single objective, we unveil a new avenue to the problem. As we consider using EMOO algorithms to solve our problem, the genetic representations found in [15], [20], [17], [16], [18], and [19] can still be deemed relevant as it is possible to use them in an EMOO algorithm.

2.3 Risk-Driven Detection of Damaged Sensors

The original RMF was proposed by Falcon et al. in [5]; the authors outlined a multimodular architecture comprising: (a) a risk feature extraction module that generates a parallel risk stream from the incoming raw data features; (b) a risk visualization module, allowing the user to monitor the system’s risk landscape in real time and (c) a risk assessment module that evaluates local and global risk dimensions of any system unit at any point in time. In [22], a response selection module was added in order to automatically determine a small set of the most promising responses to a risky event that are to be presented to an operator for further analysis. McCausland et. al. give in [23] and [24] two implementations of an RMF for maintaining perimeter coverage of an ROI. This is similar to our proposed RMF as we too are trying to preserve maximum coverage across our ROI. Falcon et al. augment the original risk feature extraction module in [25] by integrating hard-soft information fusion in order to extract risk features from a variety of sources such as radar, Automatic Identification System (AIS), intelligence reports, and historical data. More recently, the generation of an

object’s intent in a risk-aware fashion using anomaly detectors has been the subject of discussion in [26].

3 RMF for Sensor Fault Detection

In the original work [4] the robot was only deployed when the sensing holes in the ROI had formed due to sensor failure caused by battery depletion or otherwise. Here, we outline an implementation of the Risk Management Framework introduced in [5] that aims at detecting damaged sensors early on so they could be appropriately replaced with passive sensors, to maintain a maximal amount of sensor coverage over the ROI. We incorporate elements of the RMF proposed in [22] including *risk feature extraction* and the *risk assessment module*.

Risk feature extraction works by identifying the underlying data features of our system and providing a transformation into risk features used by the *risk assessment module*. This is done by creating a set of linguistic terms for each linguistic variable. Each linguistic term is modeled as a fuzzy set with a corresponding membership function. We use a Mamdani fuzzy inference system [27] for our implementation of the risk assessment module.

3.1 Data Features

We use the following four data features as the basis for our RMF:

Battery Level: A value reported from the sensor, in the range $[0, 100]$ representing the percentage of battery power remaining.

Data Fault Detection: A nominal value representing the quality of data being transmitted by a sensor. It is determined by the method proposed by Chen et al. in [28]. The nominal value has four possibilities: *possibly normal* (LG), *possibly faulty* (LT), *normal* (GD), or *faulty* (FT).

Transmission Reliability: Used to measure if the frequency at which data is periodically sent by a sensor is acceptable. This is represented by the following function:

$$f(r) = r/R_{min} \quad (1)$$

where r is the current transmission rate based on a sliding time window, in packets per second, and R_{min} is the minimum acceptable transmission rate. If $f(r) \geq R_{exp}$, where $R_{exp} = \rho R_{min}$, the expected transmission rate, and ρ is a predetermined constant, then the transmission rate is deemed to be normal. If $1 < f(r) < (\rho + 1)$ then the transmission rate is acceptable.

Physical Reliability: Represents the likelihood of a failure occurring due to the age of the sensor or its components. We represent this with the exponential failure distribution function:

$$F(t) = 1 - e^{-\lambda t} \quad (2)$$

Where t is the usage time of the sensor and λ is a constant that is determined by the make and model of the sensor.

3.2 Risk Features

Each of the data features described in Section 3.1 corresponds to a risk feature. This gives us the following risk features:

1. Battery Risk (R_1)
2. Data Fault Risk (R_2)
3. Transmission Reliability Risk (R_3)
4. Physical Reliability Risk (R_4)

Table 3: Trapezoidal Membership Functions

	Battery Risk				Transmission Reliability Risk				Physical Reliability Risk			
	A	B	C	D	A	B	C	D	A	B	C	D
HIGH	$-\infty$	0	15	30	$-\infty$	0	1	$1 + 0.25(\rho - 1)$	0.6	0.85	1	∞
MED	20	30	60	70	$1 + 0.1(\rho - 1)$	$1 + 0.6(\rho - 1)$	$1 + 0.75(\rho - 1)$	$1 + 0.85(\rho - 1)$	0.3	0.5	0.6	0.8
LOW	60	75	100	∞	$1 + 0.75(\rho - 1)$	$1 + \rho$	∞	∞	$-\infty$	0	0.33	0.5

Each of the risk features is treated as a linguistic variable with three linguistic terms: *LOW*, *MED*, and *HIGH* to indicate the level of risk that the data feature entails. Using the functions described for each data feature, we assign a risk value for each of the risk features by using trapezoidal membership functions. These functions are detailed in Table 3 for R_1 , R_3 , and R_4 . R_2 is a unique case as it is defined by four nominal values; as such, we assign the value of the risk features according to the following:

Possibly Normal (LG)	$\mapsto$ <i>LOW</i>
Normal (GD)	$\mapsto$ <i>LOW</i>
Possibly Faulty (LT)	$\mapsto$ <i>MED</i>
Faulty (FT)	$\mapsto$ <i>HIGH</i>

Using the membership values for the three numerical risk features and the nominal value for the only nominal risk feature, we are able to determine an overall risk value for each active sensor in the network. This process is outlined in the following section.

3.3 Risk Assessment Module

The assessment module of the RMF aggregates the risk features into an overall risk value for each sensor. Our RMF uses the following rules for aggregation:

- IF R_1 is *HIGH* or R_2 is *HIGH* or R_3 is *HIGH* or R_4 is *HIGH* then *SensorRisk* is *HIGH*
- IF R_1 is *MED* and R_2 is *MED* and R_3 is *MED* then *SensorRisk* is *HIGH*
- IF R_1 is *MED* or R_2 is *MED* or R_3 is *MED* or R_4 is *MED* then *SensorRisk* is *MED*
- IF R_1 is *LOW* and R_2 is *LOW* and R_3 is *LOW* and R_4 is *LOW* then *SensorRisk* is *LOW*

If a sensor belongs to *HIGH* with membership greater than 0.6 a replacement operation is triggered. Any sensor with membership to *MED* ≥ 0.6 or *HIGH* ≥ 0.6 is flagged for replacement by the mobile robot. We replace sensors with high membership to *MED* in order to reduce the number of relocation tours required by the robot to maintain network coverage. Note that some of the risk features do not increase monotonically, so threshold values for flagging must be carefully tuned to avoid replacing sensors that do not necessarily require replacement.

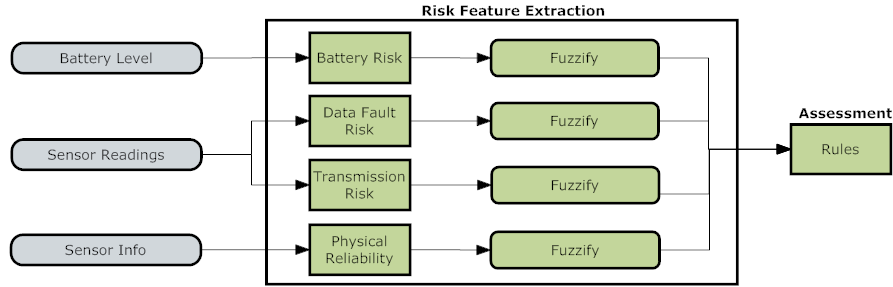


Fig. 1: An outline of the various steps of the RMF for RRASR

4 RRASR: A Multi-Objective RASR Formulation

RRASR can be expressed as a combinatorial optimization problem by representing our sensor network as a complete undirected graph $G = (V, E)$ with vertex set $V = \{v_0, \dots, v_n\}$ and edge set $E = \{e_{ij} = (v_i, v_j) | v_i, v_j \in V, v_i \neq v_j\}$ in which

each edge e_{ij} has a cost $d_{ij} > 0$ representing the Euclidean distance between v_i and v_j .

Each vertex represents either a passive sensor or a sensing hole and has an associated unitary demand q_i (1 for passive sensors and -1 for sensing holes). The base station is denoted as v_0 with $q_0 = 0$. Therefore, we can say that any other vertex either belongs to the set of passive sensors $S = \{v_i | v_i \in V, q_i = 1\}$ or the set of sensing holes $H = \{v_i | v_i \in V, q_i = -1\}$. This implies that $V = \{S \cup H \cup v_0\}$.

Each sensor has an associated battery level denoted by:

$$p_i = \begin{cases} 0 & \text{if } v_i \in H \\ -1 & \text{if } i = 0 \\ \sim U(0, 100) & \text{if } v_i \in S \end{cases}$$

There is a unique commodity (sensors) to be transported by the robot from one place to another. The robot can carry at most Q_{max} sensors and leaves the base station with an initial cargo Q_0 , $0 \leq Q_0 \leq Q_{max}$. The robot always drops the most recently picked up sensor when it reaches a sensing hole due to simulated design constraints on the robotic arm and mobile platform.

We also consider the set of all active nodes $A = \{a_0, \dots, a_m, v_0\}$ that form the WSRN deployed in the ROI, although these do not become part of the network graph G for optimization purposes. This is an essential consideration to our problem as it allows us to determine the importance of a node in the WSRN with regards to maintaining network connectivity. v_0 is specially included in A to account for those sensing holes only known to the base station.

All passive and damaged sensors $v_i \in V, 1 < i < n$ have a communication radius of R . We use the edge set $E' = \{e'_{ij} = (a_i \in A, v_j \in H)\}$ to find the active node degree g_j of each sensing hole $v_j \in H$ defined as the number of active sensors within communication distance of the sensing hole. This is formally defined as: $g_j = |\{a_i \in A : d'_{ij} \leq R\}|$ and d'_{ij} stands for the Euclidean distance between a_i and v_j where $a_i \in A, v_j \in H$.

We want to find a feasible sensor relocation trajectory, $\varphi = (v_{\gamma_1}, v_{\gamma_2}, \dots, v_{\gamma_{|\varphi|}})$ where $\gamma : [1; |V| - 1] \rightarrow \{\gamma_i\}, v \notin V \setminus \{v_0\}$ is an injective function, that starts and ends at the base station. As stated by Falcon et al. in [3], a sensor relocation trajectory is said to be feasible if it has no repeated nodes (other than the base station as its first and last element), repairs all sensor holes (i.e., drops a passive sensor at the location of each sensing hole), and never violates the robot's capacity constraint, Q_{max} . The quality of φ will be evaluated according to the following optimization objectives:

Minimize Trajectory Length

$$\sum_{e_{ij} \in \varphi} d_{ij} \quad (3)$$

Maximize Trajectory Robustness

$$\sum_{v_i \in \varphi \cap S, v_j \in \varphi \cap H} p_i g_j \delta_{ij} \quad (4)$$

Maximize Trajectory Lifetime

$$\min\{p_i\}, v_i \in \varphi \cap S \quad (5)$$

The trajectory robustness is representative of how likely a sensor hole would be to cause a disconnect in the WSN. It is expressed as a value based on the battery power of a given sensor and its score, g_j . This score is determined using the concept of *vertex separation* and the following equation:

$$g_j = 1 + \sum_{a \in A} \sum_{v_j \neq a, v_j \neq v_i, v_j \in C_{abi}} 1/|C_{abi}|, v_j \in A \quad (6)$$

where C_{abi} is the i th ab-separator [29], $a \in A$, and $b = v_0$ the depot.

The function δ_{ij} is evaluated as follows:

$$\delta_{ij} = \begin{cases} 1 & \text{if } v_i \in \varphi \cap S \text{ will be dropped off at the location of } v_j \in \varphi \cap H \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

The trajectory lifetime shows the worst passive sensor power level in a sensor relocation trajectory. We want the selected passive sensors to have enough battery power so they can sustain operational demands without being depleted in a short period of time, hence becoming damaged nodes themselves.

5 EMOO Algorithms for RRASR

This section is devoted to outlining the different components of the multiobjective optimizers under consideration to tackle the RRASR problem.

5.1 Solution Encoding

We use a modified version of a permutation-based solution representation seen in [15]. For a problem instance with a set of n nodes we will create a permutation of size $n - 1$; we do not include the base station, v_0 , in our permutation as it is assumed to be the beginning and end of every tour. Each solution will have the form $(v_1, \dots, v_n)$ to represent the order in which the nodes are visited. As not all nodes are required to be visited, we will represent the unvisited nodes by marking them as negative integers. For example, the permutation $(6, 2, -4, 1, 3, -5)$ indicates that the robot will visit nodes 6, 2, 1, and 3, in order while ignoring the others. Since RRASR must have solutions computed quickly in order to act upon them in the WSN, we adopted this representation as it ensures higher computational efficiency at the expense of requiring greater memory capacity.

5.2 Objective Functions

As we are using an MOO approach to the problem, we use (3), (4), and (5) as the fitness functions. As a result each solution is evaluated in terms of three different (and conflicting) objectives.

5.3 Population Initialization

The population is initialized by creating three heuristically determined individuals and randomly initializing the remainder of the population. The heuristically determined individuals are based on the three objective functions. For trajectory length (3) we use a nearest neighbour heuristic in which the shortest feasible edge is always taken. For trajectory robustness (4) we select the pickups and deliveries such that the best sensors are always delivered to the sensing hole with the highest score. For trajectory lifetime (5) we pickup the sensors with the highest battery levels, regardless of location. The random initialization of the remainder of the population is done by creating a permutation with no duplicate elements, with the sign (positive or negative) of each element being decided with equal probability.

5.4 Evolutionary Operators

Selection Operator We use a *binary tournament* as our selection operator. This works by randomly selecting two individuals from the population and then selecting the best of the two. As this is an MOO problem, we use the non-domination rank [30] of a given solution rather than its fitness function values. In the case that the selected solutions share the same rank either solution is chosen with equal probability.

Crossover Operator Partially-mapped crossover (PMX) [31] is used as the crossover operator. This type of crossover builds offspring solutions by selecting a subsequence from one of the parents and inserting it into the other parent, preserving the original order of as many points as possible, as demonstrated in Fig. 2.

Mutation Operator We include two mutation operators in our implementation: *insert* and *swap*. Each offspring has a chance to undergo a mutation. The *insert* operator works by selecting an element of the permutation and inserting it at another random location in the permutation and is illustrated in Figure 3. The *swap* operator works by randomly selecting two elements of the permutation and switching their values and is illustrated in Figure 4.

These mutations would have no or little effect if they were to only affect the negative elements of the permutation. As such, any element that is affected by a mutation also has its sign flipped. This causes the mutations to have a guaranteed effect on both the genotype and phenotype of the solution.

5.5 Infeasibility Handling

Solutions will be repaired if, upon evaluation, they violate any constraints. This is done so that the solution satisfies all of the problem constraints. Our proposed repair methodology can be broken down into two parts: *content repair* and *order repair*, which are completed in that order.

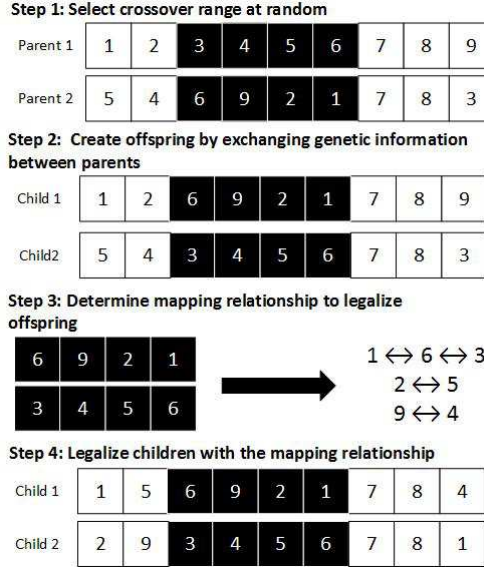
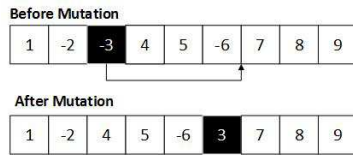
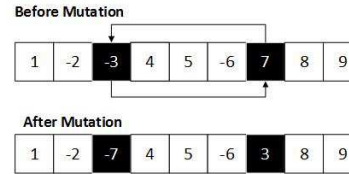


Fig. 2: Example of partially-mapped crossover

Fig. 3: Example of the *insert* mutation operatorFig. 4: Example of the *swap* mutation operator

Content Repair In the case that the solution does not contain the appropriate number of passive sensors or does not contain all of the damaged sensors we use the following procedure:

- In the case $H \subseteq \varphi$ is not true, we add the remaining elements of H to φ . We do this by making all elements belonging to H in the encoded solution positive if they are not already.
- If $|\varphi| > 2|H| - Q_0$, remove $v_i \in S \cap \varphi$ from right to left until $|\varphi| = 2|H| - Q_0$. Going from right to left prevents us from collecting passive sensors after the last sensing hole is visited. We iterate through the permutation from n to 1 and flip the sign of those v_i from positive to negative until the condition is satisfied.
- If $|\varphi| < 2|H| - Q_0$, add $v_i \in S, v_i \notin \varphi$ from left to right until $|\varphi| = 2|H| - Q_0$. Going from left to right prevents us from removing passive sensor collections that are being used as part of the solution. We iterate through the permuta-

tion from 1 to n and flip the sign of those v_i from negative to positive until the condition is satisfied.

This portion of the repair function can be seen in Figure 6.

Order Repair The ordering constraint dictates that: at all points in φ , $0 \leq Q \leq Q_{max}$ must hold. We must consider the two infeasible cases for repair:

- If $Q > Q_{max}$ at some point in φ , the offending $v_i \in \varphi \cap S$ is moved after the nearest right $v_i \in \varphi \cap H$ in φ . To do this we insert the element in which the issue appears after the rightmost element of H that appears in the permutation.
- If $Q < 0$ at some point in φ , the offending $v_i \in \varphi \cap H$ is moved after the nearest right $v_i \in \varphi \cap S$ in φ . To do this we insert the element in which the issue appears after the rightmost element of S that appears in the permutation.

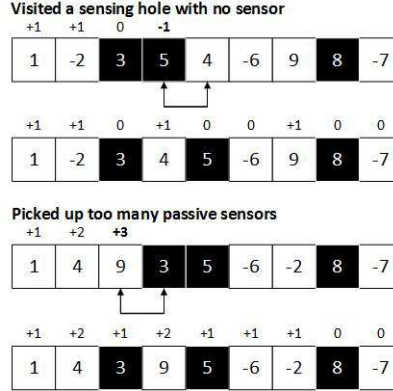


Fig. 5: Examples of sensor relocation trajectory ordering repair for $Q_{max} = 2$. In the first example the robot attempts to place a sensor when it is not carrying one. We swap the offending sensing hole with the next passive sensor. In the second example the robot attempts to collect a passive sensor when it is already carrying a full load. We swap the offending passive sensor with the next sensing hole.

This portion of the repair function is illustrated in Figure 7.

5.6 Stop Criteria

The algorithms will be terminated after they have undergone a maximum number of objective function evaluations. In a real-world deployment, it would be

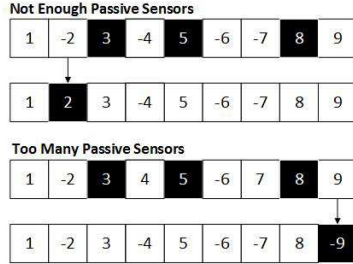


Fig. 6: In the first example there are not enough passive sensors in the relocation trajectory. We add passive sensors from right to left until we have enough. In the second example there are too many passive sensors in the relocation trajectory. We remove passive sensors from left to right.

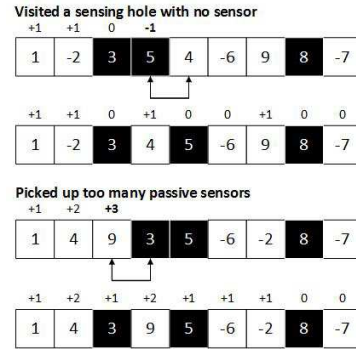


Fig. 7: Examples of sensor relocation trajectory ordering repair for $Q_{max} = 2$. In the first example the robot attempts to place a sensor when it is not carrying one. We swap the offending sensing hole with the next passive sensor. In the second example the robot attempts to collect a passive sensor when it is already carrying a full load. We swap the offending passive sensor with the next sensing hole.

more appropriate to use a time-based stop criteria as RRASR applications will generally be time-constrained. We use objective function evaluations as our stop criteria as it gives us a larger view of algorithm performance independent of time, which is used as an evaluation metric.

6 Experimental Results

This section elaborates on the empirical methodology and evaluation for our proposed MOO solution for the RRASR problem.

6.1 Experimental Setup

Synthetic Scenario Generation The scenarios used for experimentation have been generated under the control of various problem parameters such as: the total number of sensors N , ratio of sensing holes to passive sensors R_{HS} , and the distribution density of the passive sensors. In order to simulate an appropriate RRASR scenario, we generate a WSN by first mapping a grid of active sensors to a ROI. We then randomly select a subset of the active sensors to act as sensing holes. Finally, we randomly distribute the passive sensors using the *MAX_DPA* algorithm adapted from [32] with the distribution density parameter. The resulting collection of passive sensors, active sensors, and sensing holes provides the scenario used for experimentation.

We generated 110 scenarios in total: 50 scenarios for varying N from 10 to 1000 in increasing steps, 10 using varying distribution density values in *MAX_DPA* and 50 for differing R_{HS} values from 1% to 50%.

Benchmark EMOO Algorithms We examine the performance of six different EMOO algorithms on our problem. We include NSGA-II [30], which is based upon Pareto dominance and a crowding distance operator, and its successor NSGA-III [33] which follows the same framework as NSGA-II, but also uses reference planes to drive towards good solutions. SPEA2 [34] uses a strength function that is based on the number of dominated solutions of the non-dominated set. PESA-II [35] uses a region-based approach whereby the solution space is broken up into regions and solutions are chosen in order to keep the best and maintain a healthy spread across regions. The newer AGE-I [6] and AGE-II [7] algorithms are included in our benchmark group and use a method that aims to have the population model approximate the solution space called *approximation-guided evolution*. AGE-II sets itself apart from AGE-I by incorporating ε -dominance [36] and is stated to increase performance in regards to both solution quality and runtime.

Performance Metrics The performance metrics used in this study are outlined in Table 4. These are well-known MOO indicators that judge the goodness of an obtained Pareto approximation set from different angles. The reported metric values are the averages, over each synthetic RRASR scenario, of the means of 30 independent runs of an algorithm upon a scenario.

Algorithm and Parameter Configurations In order to provide a fair comparison of the algorithms we are testing, we must use the most appropriate parameters for each algorithm. To do this we conduct parametric tuning on each

Table 4: Algorithm Performance Metrics

Name	Ref.	Goal	Description
Hypervolume	[37]	Max	Represents the volume of the objective space dominated by solutions in the approximation set.
Max Pareto Front Error	[37]	Min	Represents the maximum distance from solutions in an approximation set to the nearest solution in the reference set.
Spacing	[37]	Max	Represents the spread of the Pareto approximation set.
Generational Distance	[37]	Min	Represents average distance from solutions in the approximation set to the nearest solution in the reference set.
Inverted Generational Distance	[38]	Min	Represents the average distance from solutions in the reference set to the nearest solution in the approximation set.
Elapsed Time	N/A	Min	The runtime (in seconds) of the algorithm.

algorithm to determine the best parameter values, within a set range, for each algorithm. The parameters and their ranges are listed in Table 5. We generated 240 parameters vectors using the Saltelli method [39] and each algorithm was run 15 times for each vector, with the results being averaged across all executions. We conducted this test over 14 of our synthetic data files that are representative of the set of problem instances under study.

Table 5: Algorithm Parameter Ranges

Parameter	Min. Value	Max. Value
Number of Function Evaluations	50000	200000
Population Size	50	500
Swap Mutation Rate	0.0	0.5
Insertion Mutation Rate	0.0	0.5
Crossover Rate	0.5	1.0

To determine the best vector for each algorithm we collected the metrics listed in Table 4 and gave each vector a ranking from one (best) to 240 (worst) for each metric, then took the average of those ranks. The vector with the highest average rank across all metrics was deemed the best set of parameters for a given instance file. We then took the best average vector from across all of the instance files. We did this by taking the average rank of each vector across all 14 data files. These parameter sets are used for the experiments detailed in Sections 6.2, 6.3, and 6.4 and are reported in Table 6.

Statistical Validation To confirm whether or not the differences in observed performance are statistically significant, we rely on nonparameteric tests as suggested by [40]. We use 5% significance for all tests.

Table 6: Tuned Algorithm Parameter Values

Algorithm	NFE	Pop. Size	Swap Rate	Insert Rate	Crossover Rate
AGE-I	143880	55	0.3716	0.2349	0.8628
AGE-II	172040	110	0.4966	0.1099	0.9878
NSGA-II	195545	185	0.0747	0.1255	0.9097
NSGA-III	181390	110	0.3687	0.1353	0.9878
PESA-II	195365	205	0.3843	0.4634	0.9097
SPEA2	195545	185	0.3843	0.4634	0.7651

Due to the novelty of our problem, we do not have a control algorithm. Therefore, we must first establish or reject statistically significant differences within the whole group of algorithms using the Friedman $N \times N$ procedure. In the case that a comparison yields a rejection of the null hypothesis (i.e. there exists a statistically significant difference), then a set of comparisons is done using the following post-hoc procedures: Bergmann, Nemenyi, Holm, and Schaffer [40]. We employ the Bergmann procedure given its robustness and reliable adjustment of p -value. If the Bergmann procedure does not reject a hypothesis we will defer to the other post-hoc methods.

The algorithm performance data used for the statistical validation procedure was collected at 25, 50, 75, and 100 percent of the number function evaluations across each of the scenarios.

6.2 Experiment 1: Scalability Analysis

This experiment examines the effect the size of the WSN has on the performance of the algorithms with regards to RRASR solutions. For this experiment we set the number of sensing holes to be 15 percent of the total nodes.

Figures 8, 9, and 10 show the runtime (in seconds), hypervolume, and spacing metrics respectively against the size of the sensor network. Looking at the runtime we can see that as the size of the WSN and the solution space gets bigger the runtime increases as expected. It is important to note the difference in runtime for SPEA2, and to a lesser extent AGE-II. Unlike the other algorithms, SPEA2 and AGE-II have an uncharacteristically large runtime when the solution space is small. Looking at Figure 9, we can see that this increase in runtime does not have a negative impact on solution quality. For SPEA2 this is likely the result of many duplicate solutions appearing in the archive it uses for the selection of parents, causing many unnecessary comparisons. AGE-II uses an *approximative archive* that contains only non-dominated solutions. As there are a finite number of non-dominated solutions, it is likely that they will quickly find their way into the archive causing many domination comparisons to be made to ensure that the archive only contains appropriate solutions.

In regards to overall algorithm performance we can see in Figure 9 that PESA-II and NSGA-III consistently give the best solutions, in terms of hypervolume. The only downside to these solutions is that they may not have much

variety, as we can see in Figure 10 that these algorithms underperform the others in terms of spacing. While AGE-II has the most diverse set of solutions, they are also markedly the worst. This may be due to an increased focus on exploration of the solution space preventing the solutions from converging to something good. Worth noting is that there is no apparent correlation between the size of the network and the hypervolume. This indicates that the network size alone is not a factor that allows us to predict the quality of the approximation sets.

Figure 11 shows the mean inverted generational distance achieved by each algorithm throughout its execution across all generated scenarios. It is interesting to note that while all of the algorithms are initialized with specifically tuned parameters they start and end at very similar values. Between 0 and 30 percent of function evaluations there is the largest disparity between algorithms, but after that they seem to converge towards a small set of desirable distance values.

Table 7: Experiment 1 - Friedman $N \times N$ Test Results

Metric	Algorithm Rank						p -value
	NSGA-II	NSGA-III	SPEA2	PESA-II	AGE-I	AGE-II	
Hypervolume	3.3175	2.0125	4.615	2.015	4.19	4.85	2.0159E-10
Spacing	4.7575	4.04	2.005	5.14	3.5225	1.535	1.9077E-10
Max Pareto Front Error	4.3375	3.1025	3.56	2.215	3.91	3.875	1.0060E-10
Generational Distance	3.2475	1.955	5.06	1.89	3.6175	5.23	1.9471E-10
Inverted Generational Distance	3.0375	1.86	4.975	1.97	4.0725	5.085	2.2732E-10
Runtime	2.81	3.335	4.27	1.19	4.005	5.39	2.2589E-10

Table 7 gives an overview of the statistical analysis carried out for each of the recorded metrics. The small p -values indicate that the results of the analysis are statistically significant for the group of algorithms under each metric. For each metric the following null hypothesis are not rejected:

Hypervolume: SPEA2 vs. AGE-II, NSGA-III vs. PESA-II

Max Pareto Front Error: AGE-I vs. AGE-II, SPEA2 vs. AGE-II, SPEA2 vs. AGE-I, NSGA-II vs. AGE-I, NSGA-III vs. SPEA2, NSGA-II vs. AGE-II

Generational Distance: NSGA-III vs. PESA-II, SPEA2 vs. AGE-II, NSGA-II vs. AGE-I

Inverted Generational Distance: NSGA-III vs. PESA-II, SPEA2 vs. AGE-II

Runtime: SPEA2 vs. AGE-I

It is important to note that while the metrics show that either NSGA-III or PESA-II is the best, except for max Pareto front error and runtime, none

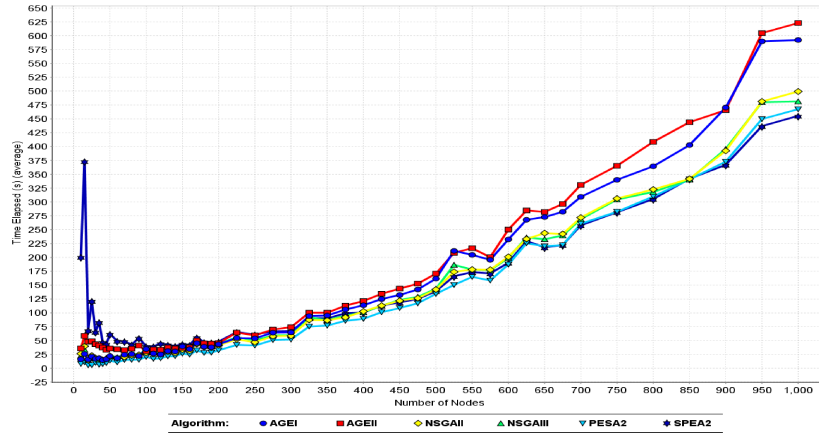


Fig. 8: Final execution results for time elapsed, in seconds, across all generated scenarios for the scalability analysis experiment

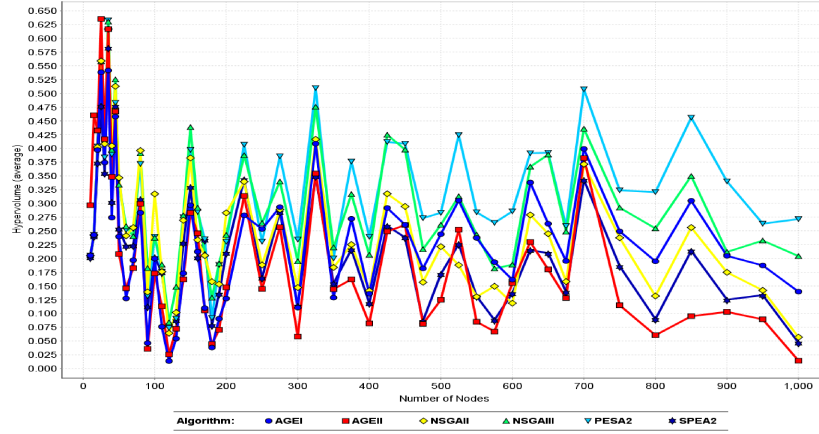


Fig. 9: Final execution results for hypervolume, across all generated scenarios for the scalability analysis experiment

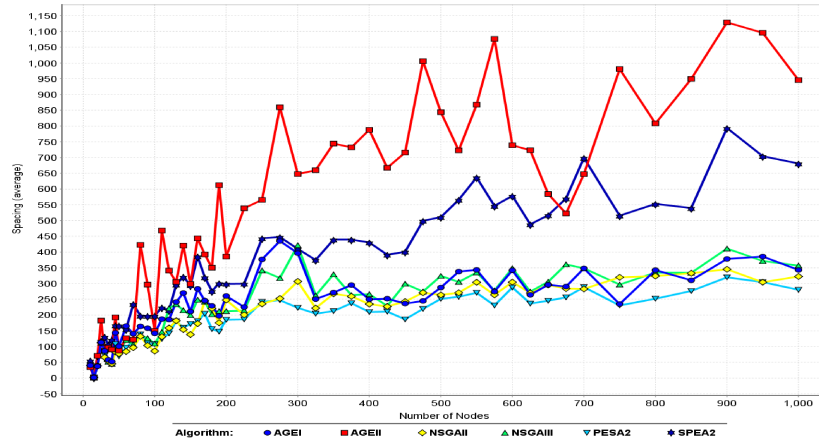


Fig. 10: Final execution results for spacing, across all generated scenarios for the scalability analysis experiment

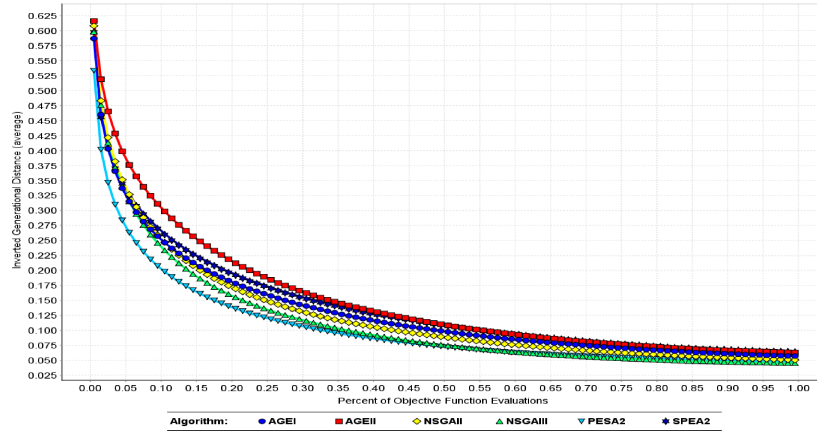


Fig. 11: Average results, across all generated scenarios, for the inverted generational distance for the scalability analysis experiment

of the *post-hoc* procedures are able to reject the NSGA-III vs. PESA-II null hypothesis. Therefore, we are unable to definitively say whether one algorithm has performed better than the other. However, we are able to state that both NSGA-III and PESA-II outperformed the other algorithms.

6.3 Experiment 2: Inflicted Damage Analysis

In this experiment we examine the performance of the selected EMOO algorithms in a WSN comprised of 200 static nodes where an increasing number of them are labelled for replacement, e.g., in case of a malicious attack.

Concerning runtime, we can see a similar trend in Figure 12 as the one depicted in Experiment 1. As the solution space increases in size we observe a smooth linear dependency between the percentage of nodes flagged for replacement and the time taken to compute a solution. SPEA2 and, to a much lesser extent, AGE-II suffers in terms of runtime when the solution space is small. This is likely due to the same reasons mentioned in our runtime discussion in Experiment 1.

Figure 13 shows that all of the algorithms perform similarly when a lower percentage of sensors needs to be replaced, but as that percentage increases the disparity of performance between algorithms appears. At damage levels above 20 percent it can be seen that NSGA-III and PESA-II identify themselves as frontrunners for performance. This is likely due to the region-based selection method used by PESA-II and the method of selecting reference lines to drive intelligent exploration used by NSGA-III.

The results in terms of spacing are shown in Figure 14. We can see that AGE-II and SPEA2 perform the best in terms of the variety of solutions that they present. The issue is that the quality of those solutions is not as high as those

produced by other algorithms; this is especially true of AGE-II when comparing its results in Figures 13 and 14. All of the other algorithms seem to have a similar amount of diversity among their resulting solutions which is interesting given their various selection methods.

Figure 15 shows a snapshot of algorithm performance at various percentages of the execution budget, expressed as an average of the number of objective function evaluations across all scenarios. We can see that the majority of difference in performance comes before 30 percent of function evaluations. It is interesting that each of the algorithms converge at close to the same narrow interval of inverted generational distance values. These results are near identical to those observed in Experiment 1.

Table 8: Experiment 2 - Friedman $N \times N$ Test Results

Metric	Algorithm Rank						p -value
	NSGA-II	NSGA-III	SPEA2	PESA-II	AGE-I	AGE-II	
Hypervolume	3.8375	2.0425	4.55	1.8825	4.035	4.6525	1.6168E-10
Spacing	4.485	4.465	1.99	5.285	3.595	1.18	2.8214E-10
Max Pareto Front Error	4.1575	2.905	3.4775	2.44	3.16	4.86	9.6999E-11
Generational Distance	3.075	1.9125	4.9	2.0475	3.24	5.825	2.6979E-10
Inverted Generational Distance	3.65	1.905	4.82	1.885	3.845	4.895	1.9804E-10
Runtime	3.075	3.875	5.825	1.03	3.315	3.88	2.0301E-10

Table 8 gives an overview of the statistical analysis found for each of the recorded metrics. The small p -values indicate that the results of the analysis are statistically significant for the group of algorithms under each metric. For each metric the following null hypothesis are not rejected by any *post-hoc* procedure:

Hypervolume: SPEA2 vs. AGE-II, NSGA-III vs. PESA-II, NSGA-II vs. AGE-I

Spacing: NSGA-II vs. NSGA-III

Max Pareto Front Error: NSGA-III vs. AGE-I, SPEA2 vs. AGE-I

Generational Distance: NSGA-III vs. PESA-II, NSGA-II vs. AGE-I

Inverted Generational Distance: NSGA-III vs. PESA-II, SPEA2 vs. AGE-II, NSGA-II vs. AGE-I

Runtime: NSGA-III vs. AGE-II, NSGA-II vs. AGE-I

The important hypotheses that are not rejected are NSGA-III vs. PESA-II for any metric. As these two algorithms are the best performing overall, it is difficult to identify a single algorithm that is superior. Additional experimentation and

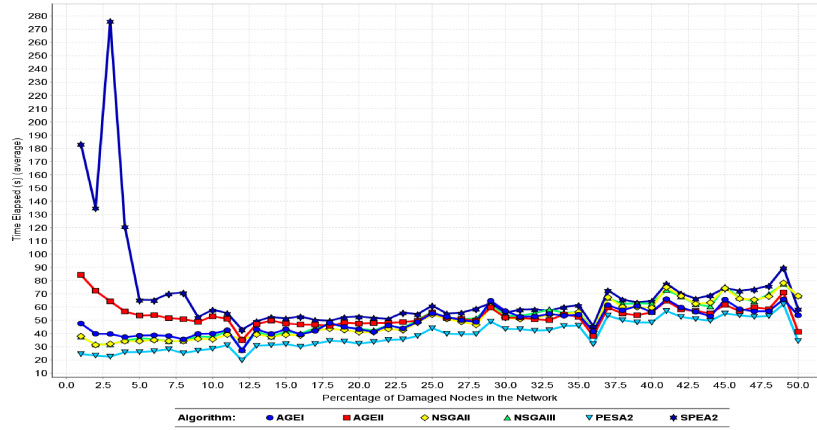


Fig. 12: Final execution results for time elapsed, in seconds, across all generated scenarios for the inflicted damage analysis experiment

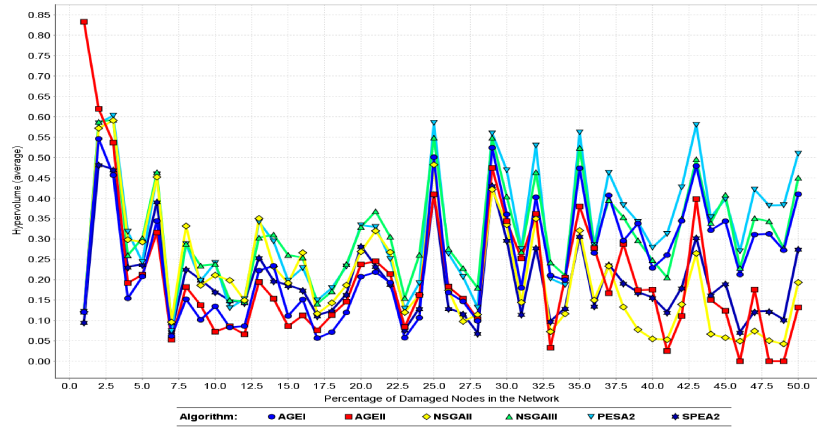


Fig. 13: Final execution results for hypervolume, across all generated scenarios for the inflicted damage analysis experiment

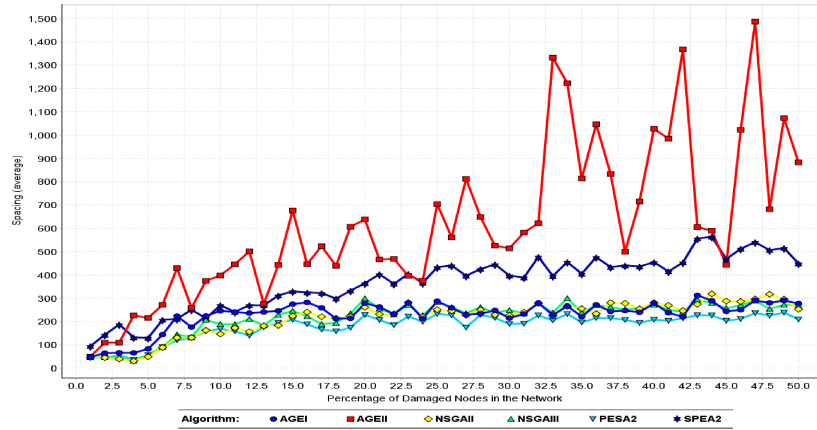


Fig. 14: Final execution results for spacing, across all generated scenarios for the inflicted damage analysis experiment

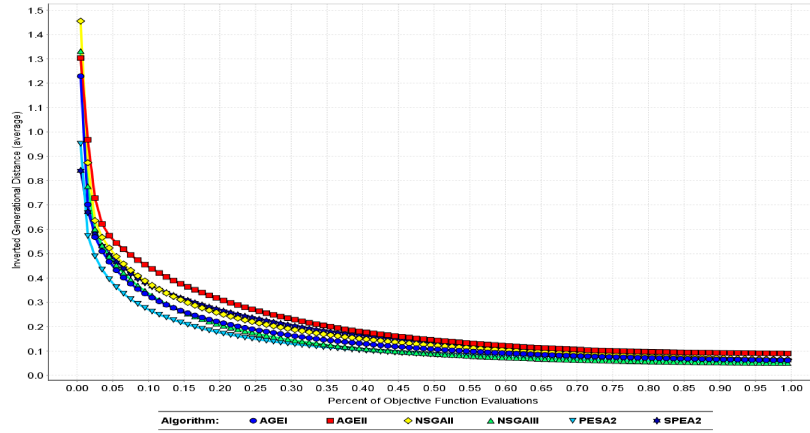


Fig. 15: Average results, across all generated scenarios, for the inverted generational distance for the inflicted damage analysis experiment

executions would be needed to identify definitively which algorithm performs better in the RRASR problem space.

6.4 Experiment 3: Network Density Analysis

In this experiment we examine the effects of WSN topology on algorithm performance. To do this we generated solutions using the *MAX_DPA* method from [32]. We use distribution values ranging from 0 to 10 representing the minimum number of neighbors a sensor can have. We are interested in assessing the behaviour of the EMOO algorithms in both sparse and dense networks.

Figure 16 shows the runtime for each of the algorithms. This indicator remains similar across all network density values, therefore any disparity between algorithms is also highlighted. The same can be said for Figure 17 where we examine the hypervolume of the EMOO algorithms at each network density value. We again see that the algorithms tend to perform similarly across each network density value.

When considering spacing, Figure 18 shows a different trend for AGE-II. While all the other algorithms tend to perform the same across node density values, AGE-II sets itself apart by having an upward trend as the node density value increases. This is an important find as it may indicate that the topology of the WSN has an effect on algorithm performance rather than just the size of the solution space, as demonstrated by Experiments 1 and 2. This is especially interesting as the EMOO algorithms have no way of noticing differences in network topology.

Looking at the mean inverted generational distance across all executions in 19 we see results much the same as Experiments 1 and 2. The algorithms show difference up to about 30 percent of function evaluations and then trend towards

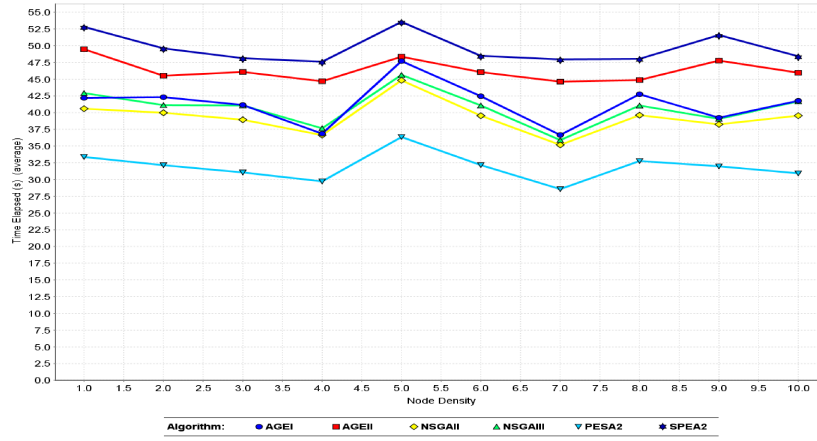


Fig. 16: Final execution results for time elapsed, in seconds, across all generated scenarios for the network density analysis experiment

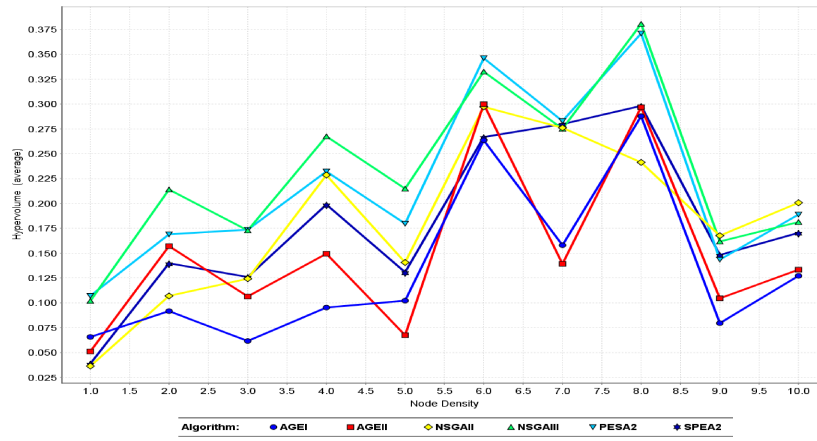


Fig. 17: Final execution results for hypervolume, across all generated scenarios for the network density analysis experiment

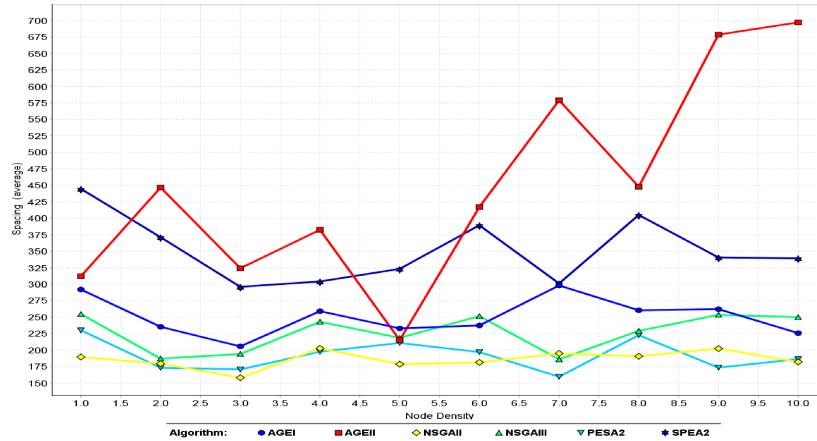


Fig. 18: Final execution results for spacing, across all generated scenarios for the network density analysis experiment

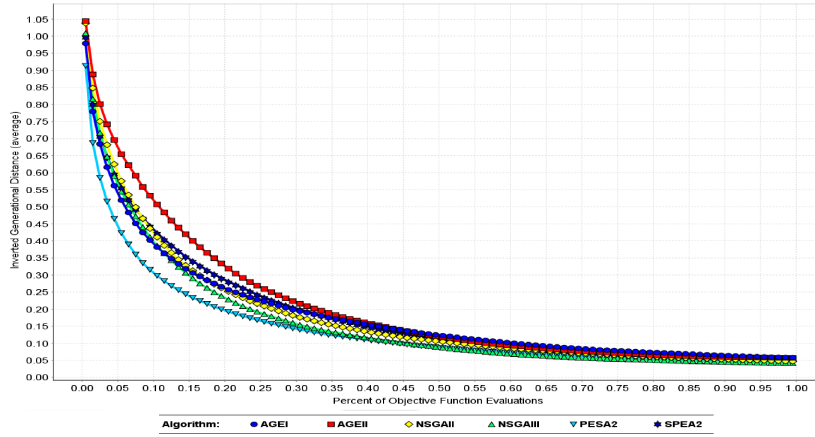


Fig. 19: Average results, across all generated scenarios, for the inverted generational distance for the network density analysis experiment

the same point. This shows that each algorithm had managed to discover at least a few solutions that can be considered objectively good.

Table 9: Experiment 3 - Friedman $N \times N$ Test Results

Metric	Algorithm Rank						p -value
	NSGA-II	NSGA-III	SPEA2	PESA-II	AGE-I	AGE-II	
Hypervolume	3.4875	1.775	3.925	2	5.0875	4.725	8.1971E-11
Spacing	5.45	3.9	1.8	5.025	3.3	1.525	1.1685E-10
Max Pareto Front Error	4.7	3.375	2.7	2.2	4.375	3.65	5.0325E-10
Generational Distance	2.75	1.675	4.925	1.75	3.9	6	9.3480E-10
Inverted Generational Distance	3.45	1.8	4.5	1.625	4.8	4.825	6.2807E-11
Runtime	2.1	3.25	6	1	3.675	4.975	1.0208E-10

An overview of the statistical analysis for Experiment 3 is given in Table 9. We can see by the very small p -values that all of the results are statistically significant at the 5% level. For each metric the following null hypothesis are not rejected by any *post-hoc* procedure:

Hypervolume: NSGA-III vs. PESA-II, AGE-I vs. AGE-II, NSGA-II vs. SPEA2, SPEA2 vs. AGE-II

Spacing: SPEA2 vs. AGE-II, NSGA-II vs. PESA-II, NSGA-III vs. AGE-I

Max Pareto Front Error: NSGA-III vs. AGE-II, NSGA-II vs. AGE-I, SPEA2 vs. PESA-II, NSGA-III vs. SPEA2, AGE-I vs. AGE-II, SPEA2 vs. AGE-II, NSGA-III vs. AGE-I, NSGA-II vs. AGE-II
Generational Distance: NSGA-III vs. PESA-II
Inverted Generational Distance: AGE-I vs. AGE-II, NSGA-III vs. PESA-II, SPEA2 vs. AGE-I, SPEA2 vs. AGE-II
Runtime: NSGA-III vs. AGE-I

We can notice some cases a null hypothesis with regards to the best performing algorithm is not rejected. This is the case for NSGA-II vs. PESA-II for hypervolume, generational distance, and inverted generational distance, for SPEA2 vs. AGE-II for spacing, and SPEA2 vs. PESA-II for max Pareto front error. In these cases we cannot definitively say that one algorithm has performed the best, only that some have performed better than others. We can also see that PESA-II always performed the best in terms of runtime, and that SPEA2 always performed the worst. AGE-II also always performed the worst in terms of generational distance.

7 Conclusions

In this chapter we have further defined the RRASR problem as a more robust version of the RASR problem, so that we can generate more reliable trajectories for a mobile robot. We also proposed a risk-centric modeling of the sensor units in order to identify problematic sensors so that they can be pro-actively replaced in order to maximize network coverage. RRASR was updated from its original version so that its trajectory robustness function was more accurate in identifying nodes that were important to network connectivity. Six state-of-the-art EMOO algorithms were evaluated with respect to RRASR. The three experiments examined algorithm performance over networks of various sizes, inflicted damage, and sensor densities. Our results show that PESA-II and NSGA-III performed better than the other algorithms for all experiments. It is interesting to note the difference in results from the original work [4] which is likely due to the inclusion of parametric tuning. AGE-I and AGE-II were surprising in their poor performance given their success against the other algorithms using reference problems. As future work we would like to expand the problem to include multiple robot solutions.

References

1. J. Yick, B. Mukherjee, and D. Ghosal, “Wireless sensor network survey,” *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.
2. R. Falcon, *Towards Fault Reactiveness in Wireless Sensor Networks with Mobile Carrier Robots*. PhD thesis, University of Ottawa, Ottawa, Ontario, Canada, 2012.
3. 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 *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, (Barcelona, Spain), pp. 4326–4333, 2010.

4. B. Desjardins, R. Falcon, R. Abielmona, and E. Petriu, "A multi-objective optimization approach to reliable robot-assisted sensor relocation," in *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pp. 956–964, IEEE, 2015.
5. R. Falcon, A. Nayak, and R. Abielmona, "An evolving risk management framework for wireless sensor networks," in *Computational Intelligence for Measurement Systems and Applications (CIMS), 2011 IEEE International Conference on*, pp. 1–6, IEEE, 2011.
6. K. Bringmann, T. Friedrich, F. Neumann, and M. Wagner, "Approximation-guided evolutionary multi-objective optimization," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, p. 1198, 2011.
7. M. Wagner and F. Neumann, "A fast approximation-guided evolutionary multi-objective algorithm," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pp. 687–694, ACM, 2013.
8. M. Lian-Ming and D. Xi-Li, "A novel ant colony system for solving the one-commodity traveling salesman problem with selective pickup and delivery," in *Natural Computation (ICNC), 2012 Eighth International Conference on*, pp. 1096–1101, IEEE, 2012.
9. R. Falcon, X. Li, A. Nayak, and I. Stojmenovic, "A Harmony-Seeking Firefly Swarm to the Periodic Replacement of Damaged Sensors by a Team of Mobile Robots," in *2012 IEEE International Conference on Communications (ICC)*, (Ottawa, Canada), pp. 6436–6440, June 2012.
10. K. Magklara, D. Zorbas, and T. Razafindralambo, "Node discovery and replacement using mobile robot," in *Ad Hoc Networks*, pp. 59–71, Springer, 2013.
11. Y. Wang, A. Barnawi, R. F. De Mello, and I. Stojmenovic, "Localized ant colony of robots for redeployment in wireless sensor networks," *Multi-Valued Logic & Soft Computing*, vol. 23, pp. 35–51, 2014.
12. G. Fletcher, X. Li, A. Nayak, and I. Stojmenovic, "Randomized robot-assisted relocation of sensors for coverage repair in wireless sensor networks," in *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*, pp. 1–5, IEEE, 2010.
13. Y. Miao and W. Yu-Ping, "Coverage repair strategies for wireless sensor networks using mobile actor based on evolutionary computing," *Bulletin of Electrical Engineering and Informatics*, vol. 3, no. 3, pp. 213–222, 2014.
14. H. Li, A. Barnawi, I. Stojmenovic, and C. Wang, "Market-based sensor relocation by robot team in wireless sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 22, pp. 259–280, 2014.
15. X.-L. Liao and C.-K. Ting, "An evolutionary approach for the selective pickup and delivery problem," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pp. 1–8, IEEE, 2010.
16. Y.-H. Huang and C.-K. Ting, "Genetic algorithm with path relinking for the multi-vehicle selective pickup and delivery problem," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pp. 1818–1825, June 2011.
17. X.-L. Liao and C.-K. Ting, "Evolutionary algorithms using adaptive mutation for the selective pickup and delivery problem," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pp. 1–8, IEEE, 2012.
18. B. P. Bruck, A. G. dos Santos, and J. E. C. Arroyo, "Hybrid metaheuristic for the single vehicle routing problem with deliveries and selective pickups," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pp. 1–8, IEEE, 2012.
19. B. P. Bruck, A. Santos, and J. Arroyo, "An evolutionary algorithm and a variable neighborhood descent algorithm for the single vehicle problem with deliveries and

- selective pickups,” *Proceedings of the 2012 CLAIO/SBPO (Rio de Janeiro, Brazil, 2012)*.
20. C.-K. Ting and X.-L. Liao, “The selective pickup and delivery problem: Formulation and a memetic algorithm,” *International Journal of Production Economics*, vol. 141, no. 1, pp. 199–211, 2013.
 21. B. P. Bruck and A. dos Santos, “Hybrid approach for the multiple vehicle routing problem with deliveries and selective pickups,” in *Hybrid Intelligent Systems (HIS), 2012 12th International Conference on*, pp. 265–270, Dec 2012.
 22. R. Falcon and R. Abielmona, “A response-aware risk management framework for search-and-rescue operations,” in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pp. 1–8, IEEE, 2012.
 23. J. McCausland, G. Di Nardo, R. Falcon, R. Abielmona, V. Groza, and E. Petriu, “A proactive risk-aware robotic sensor network for critical infrastructure protection,” in *Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), 2013 IEEE International Conference on*, pp. 132–137, IEEE, 2013.
 24. J. McCausland, R. Abielmona, R. Falcon, A.-M. Cretu, and E. Petriu, “Auction-based node selection of optimal and concurrent responses for a risk-aware robotic sensor network,” in *Robotic and Sensors Environments (ROSE), 2013 IEEE International Symposium on*, pp. 136–141, IEEE, 2013.
 25. R. Falcon, R. Abielmona, S. Billings, A. Plachkov, and H. Abbass, “Risk management with hard-soft data fusion in maritime domain awareness,” in *Computational Intelligence for Security and Defense Applications (CISDA), 2014 Seventh IEEE Symposium on*, pp. 1–8, IEEE, 2014.
 26. R. Falcon, R. Abielmona, and S. Billings, “Risk-driven intent assessment and response generation in maritime surveillance operations,” in *Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2015 IEEE International Inter-Disciplinary Conference on*, pp. 151–157, IEEE, 2015.
 27. E. H. Mamdani and S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller,” *International journal of man-machine studies*, vol. 7, no. 1, pp. 1–13, 1975.
 28. J. Chen, S. Kher, and A. Somani, “Distributed fault detection of wireless sensor networks,” in *Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, pp. 65–72, ACM, 2006.
 29. M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*. Radarweg 29, 1043 NX Amsterdam, Netherlands: Elsevier, 2 ed., 2 2004.
 30. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, Apr. 2002.
 31. D. E. Goldberg and R. Lingle, “Alleles, loci, and the traveling salesman problem,” in *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, vol. 154, Lawrence Erlbaum, Hillsdale, NJ, 1985.
 32. F. A. Onat, I. Stojmenovic, and H. Yanikomeroglu, “Generating random graphs for the simulation of wireless ad hoc, actuator, sensor, and internet networks,” *Pervasive and Mobile Computing*, vol. 4, no. 5, pp. 597–615, 2008.
 33. K. Deb and H. Jain, “An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints,” *Evolutionary Computation, IEEE Transactions on*, vol. 18, pp. 577–601, Aug 2014.
 34. E. Zitzler, M. Laumanns, L. Thiele, E. Zitzler, E. Zitzler, L. Thiele, and L. Thiele, “SPEA2: Improving the strength pareto evolutionary algorithm,” 2001.

35. D. W. Corne, N. R. Jerram, J. D. Knowles, M. J. Oates, *et al.*, “PESA-II: Region-based selection in evolutionary multiobjective optimization,” in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2001)*, pp. 283–290, Morgan Kaufmann Publishers, 2001.
36. M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, “Combining convergence and diversity in evolutionary multiobjective optimization,” *Evolutionary computation*, vol. 10, no. 3, pp. 263–282, 2002.
37. C. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007.
38. M. R. Sierra and C. A. C. Coello, “Improving pso-based multi-objective optimization using crowding, mutation and-dominance,” in *Evolutionary multi-criterion optimization*, pp. 505–519, Springer, 2005.
39. A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.
40. J. Derrac, S. García, D. Molina, and F. Herrera, “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms,” *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

Reliable Multiple Robot-Assisted Sensor Relocation using Multi-Objective Optimization

5.1 Preamble

This chapter contains the conference proceeding “*Reliable Multiple Robot-Assisted Sensor Relocation using Multi-Objective Optimization*” [3]. This work expands on the previous two by modifying the RRASR problem to account for the multiple-robot case. In doing so, a fourth objective function is added to account for the load balancing among the participating robots. The other problem objectives are also modified to account for this. Additionally, the formulation of the genetic algorithm building blocks is also revisited for the multi-robot case. The algorithms used in [2] are evaluated against this new problem using the set of metrics used in [1]. We also test a composite algorithm, called

LifeCycle [79], which combines various algorithms to perform a composite search in an attempt to avoid stagnation in the search undertaken by any algorithm. There are three experiments conducted: the effect of network size on the problem, and the performance of the algorithms on the problem with varying degrees of inflicted damage on the network, and the effect of the number of robots on the solutions. We find that SPEA2 outperforms all of the other examined algorithms and that PESA-II has a drastically increased runtime with the addition of the fourth objective. We also find that LifeCycle performs worse than our best performing algorithm even though SPEA2 is a component of the LifeCycle that is examined.



RightsLink®

[Home](#)
[Create Account](#)
[Help](#)


Title: A multi-objective optimization approach to reliable Robot-Assisted Sensor Relocation

Conference Proceedings: 2015 IEEE Congress on Evolutionary Computation (CEC)

Author: Benjamin Desjardins; Rafael Falcon; Rami Abielmona; Emil Petriu

Publisher: IEEE

Date: 25-28 May 2015

Copyright © 2015, IEEE

[LOGIN](#)

If you're a [copyright.com](#) user, you can login to RightsLink using your copyright.com credentials. Already a [RightsLink user](#) or want to [learn more?](#)

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

[BACK](#)
[CLOSE WINDOW](#)

Copyright © 2016 [Copyright Clearance Center, Inc.](#) All Rights Reserved. [Privacy statement](#). [Terms and Conditions](#).
Comments? We would like to hear from you. E-mail us at customercare@copyright.com

Reliable Multiple Robot-Assisted Sensor Relocation using Multi-Objective Optimization

Benjamin Desjardins, Rafael Falcon, Rami Abielmona and Emil Petriu

Abstract

Wireless sensor networks provide a way to monitor a region of interest. Incorporating a robot into the sensor network provides a basis for other types of functionality to be added. One possibility is the replacement of damaged sensors with excess sensors within the wireless sensor network. This scenario has been defined as the “*Robot-Assisted Sensor Relocation*” (RASR) problem and focused only on minimizing the length of the trajectory taken by the robot. RASR has been recently expanded on as a multi-objective optimization (MOO) problem to examine a more realistic scenario by considering the reliability and placement location of the passive sensors used for replacement; this new problem is termed “*Reliable Robot-Assisted Sensor Relocation* (RRASR). In this paper, the possibility of multiple robots servicing the sensor network is considered and the RRASR problem formulation is modified accordingly. In addition, load balancing of robots by adding an objective function to the MOO representation is included. We refer to this multi-robot version as *Reliable Multiple Robot-Assisted Sensor Relocation*. The performance of six state-of-the-art evolutionary MOO algorithms using sensor networks of varying sizes and inflicted damage levels is examined.

Index Terms

evolutionary multi-objective optimization; genetic algorithms; wireless sensor and robot networks; robotics; wireless sensor networks; sensor relocation

I. INTRODUCTION

A wireless sensor network (WSN) can be described as a collection of wireless sensors deployed in a region of interest (ROI) [1]. A wireless sensor is a piece of hardware equipped with some form of wireless communication, one or more monitoring devices, and an energy source (e.g. battery) [1]. These sensors are deployed in ROIs in order to gather low-level information used in high-level domains. These sensors must be deployed so that there are no sensing holes in order to provide complete and uninterrupted coverage of a ROI.

When a robot or group of robots is added to a WSN in order to perform some sensing and/or actuation task related to the WSN it is termed a Wireless Sensor and Robot Network (WSRN) [2]. Whereas sensors may be limited in their capabilities, robots are flexible, generally mobile, and have fewer or no resource limitations allowing them to optimize, assist, and/or maintain the sensor network. WSRNs fall into two categories: *robot-assisted WSNs* and *robot-dependent WSNs* [3]. In the first case, the robot(s) are not required by the system, but enhance its functionality and/or performance. In the second case, the robot(s) are required for the system to function.

A specific WSRN application, called *Robot-Assisted Sensor Relocation* (RASR), also referred to as “*carrier-based coverage repair*” [4] in the literature, involves one or more robots that are responsible for the maintenance of the WSN. In RASR, the robot(s) remain on standby at a common base station and are launched into the field to relocate sensors in the network in order to maximize performance. RASR assumes that there are excess sensors in the ROI. These excess sensors need not collect information for maximum coverage and will henceforth be referred to as passive sensors. Passive sensors are used by the robot(s) to replace damaged sensors. The robot(s) will leave the base station to reconfigure sensors when enough damage has been caused to the network.

We consider an RASR problem with full ROI coverage, as well as extra, unused, sensors that remain in a passive state. Desjardins et al. proposed a solution to this problem in [5] called *Reliable Robot-Assisted Sensor Relocation* (RRASR). RRASR differs from RASR in that a multi-objective optimization (MOO) approach is taken where, in addition to the trajectory length, the battery level of replacement sensors is considered as well as sensor placement with regards to network connectivity. We extend on RRASR by proposing a multi-robot variant of the problem called *Reliable Multiple Robot-Assisted Sensor Relocation* (RMRASR).

We make the following contributions in this paper: (1) we review relevant works in the literature from both sensor relocation and optimization perspectives; (2) we extend the RRASR formulation [5] to account for multiple robots; (3) we introduce a load balancing objective function to try and equally divide the work done by each robot; (4) we provide empirical results and analysis with regards to the performance of six current evolutionary multi-objective optimization (EMOO) algorithms in

B. Desjardins and E. Petriu are with the School of Electrical Engineering & Computer Science, University of Ottawa, Ottawa ON, K1N 6N5 Canada {bdesj038, petriu}@uottawa.ca

R. Falcon and R. Abielmona are with the Research & Engineering Division, Larus Technologies Corporation, Ottawa ON, K1P 5V5 Canada {rafael.falcon, rami.abielmona}@larus.com

experiments dealing with WSNs of various sizes and inflicted damage levels; (5) we include and test an implementation of the LifeCycle [6] method based on three of our algorithms under consideration.

TABLE I
RELATED OPTIMIZATION LITERATURE

Ref.	Optimization Problem	Optimization Algorithm	Load Balancing	Notes
[7]	VRP	GA with local search as mutation	Max load per trip	Single vehicle with many trips
[8]	VRP with simultaneous delivery and pickup points	TSP solution algorithm	Geographic grouping	Solves each geographic group using a TSP algorithm
[9]	MVRPDSP	Hybrid Metaheuristic using clustering and subtour elimination	Geographic grouping	Allows pickup and delivery from same location.
[10]	Capacitated VRP	Hybrid GA using local search	Centroids	Dual population used for GA
[11]	Capacitated VRP	Particle Swarm Optimization	Evolved Centroid	Uses weighting in permutation to determine ordering
[12]	VRP	GA	None	Groups nodes by vehicle then uses a nearest neighbour algorithm to determine the best route.
[13]	MVRP	GA	k -means clustering	Problem is broken up into sub-problems by clustering and each is solved individually.
[14]	Multi-vehicle SPDP	GA with Path Relinking	Solution encoding	All vehicles are used in each instance.
[15]	VRP with time windows	Basic Pareto-based multi-objective GA	None	Vehicles have a capacity constraint Vehicles added to solution as constraints are violated

The rest of the paper is structured as follows. Section II briefly reviews relevant works. Section III formalizes the RMRASR problem and Section IV details the algorithmic building blocks for the EMOO schemes under consideration. Section V is concerned with the empirical evaluation of our proposed methodology. Finally, Section VI concludes the paper.

II. RELATED WORK

A. Robot-Assisted Sensor Relocation

RASR has been studied using various methods, including single and multi-robot variants. Some works employ a localized approach in which all decisions are made by the robots using information from the sensors in their vicinity. This can be seen in the work by Li et al. [16]. The centralized approach has a central server that determines the appropriate sensor relocation trajectory for the robot to take, as reported in [17].

Our approach extends the MOO-based approach to [17] that was presented in [5] to the multi-robot scenario. We add a fourth objective function that tries to balance the sensor relocation workload among the participating robots.

B. Related Optimization Problems

The generalized version of RASR, in which pickup and delivery locations can store more than a single item, has been studied as the *selective pickup and delivery problem* (SPDP) [18] and the *Single Vehicle Routing Problem with Deliveries and Selective pickups* (SVRPDSP) [9]. We examine the multi-vehicle versions of this problem: *Multiple Vehicle Routing Problem with Delivery and Selective Pickups* (MVRPDSP) and its associated generalizations termed the *Multiple Vehicle Routing Problem* (MVRP). A summary of related problems can be seen in Table I. Since the *vehicle routing problem* (VRP) and *travelling salesman problem* (TSP) are known to be NP-hard, we can also state that RASR and MVRP are as well.

There are many slight variations on the VRP and many proposed solution methods. Prins [7] proposes a VRP solution that uses GA with a local search as mutation. This solution aims to minimize the total distance travelled, but may require many trips. Any solution that incorporates many trips can be abstracted to be one vehicle per trip, making it an MVRP problem. The MVRP with simultaneous delivery and pick-up points is first mathematically modelled by Min [8], and uses a TSP algorithm to solve sub-problems created by grouping the nodes geographically. This geographic grouping is also used by Bruck and dos Santos [9] to solve MVRP with delivery and selective pick-ups, which is similar to our proposed problem. The works in [10] and [11] both use centroid methods for grouping points geographically. [10] uses a hybrid GA to solve the problem and [11] uses particle swarm optimization. Baker [12] examines the standard VRP using a genetic algorithm (GA) which groups nodes by vehicle and then applies a nearest neighbour algorithm to create routes for each vehicle. The MVRP is solved using a GA by Nallusamy [13]. This solution uses k -means clustering to group nodes and then solves each sub-problem separately. Huang and Ting [14] examine the MVRPDSP and use a GA with path relinking in order to improve their solution quality. They distribute the work among vehicles by including the number of locations serviced by each vehicle in their solution permutation. A multi-objective approach to the VRP with time windows, a specific VRP scenario, is proposed in [15]. They minimize both

route length and total number of vehicles. Our work differs from the previously mentioned by approaching the problem from a multi-objective perspective.

III. RMRASR: A FORMULATION OF MULTI-ROBOT RRASR

RMRASR is expressed as a combinatorial optimization problem by representing our WSN as a complete undirected graph $G = (V, E)$ with vertex set $V = \{v_0, \dots, v_n\}$ and edge set $E = \{e_{ij} = (v_i, v_j) | v_i, v_j \in V, v_i \neq v_j\}$ in which each edge e_{ij} has a cost $d_{ij} > 0$ representing the Euclidean distance between v_i and v_j .

Each vertex represents either a passive sensor or a sensing hole caused by a damaged sensor and has an associated unitary demand q_i (-1 for passive sensors and 1 for sensing holes). The base station is denoted as v_0 with $q_0 = 0$; in order to represent the multi-robot nature of our problem we include $b - 1$, where b is the number of robots, additional vertices to G to represent duplicate base stations. These duplicate base stations are used to partition the solution vector into a distinct trajectory for each of the b robots. The set of depot vertices is represented by $B = \{v_i | q_i = 0, v_i \in V\}$. Therefore, we can say that any other vertex either belongs to the set of passive sensors $S = \{v_i | v_i \in V, q_i = -1\}$ or the set of sensing holes $H = \{v_i | v_i \in V, q_i = 1\}$. This implies that $V = \{S \cup H \cup B\}$.

Each sensor has an associated battery level denoted by:

$$p_i = \begin{cases} 0 & \text{if } v_i \in H \\ -1 & \text{if } v_i \in B \\ \sim U(0, 100) & \text{if } v_i \in S \end{cases}$$

There is a unique commodity (sensors) to be relocated by a robot. Each robot can carry at most Q_{max} sensors and leaves the base station with an initial cargo Q_0 , $0 \leq Q_0 \leq Q_{max}$. A robot always drops the most recently collected sensor when it reaches a sensing hole due to simulated design constraints on the robotic arm and mobile platform.

We also consider the set of all active sensors $A = \{a_0, \dots, a_m, v_0\}$ that form the WSN deployed in the ROI. This allows us to determine the importance of a sensor in the WSN with regards to maintaining network connectivity. v_0 is specially included in A to account for sensing holes only known to the base station.

All passive and damaged sensors $v_i \in V, 1 < i < n$ have a communication radius of R . We use the edge set $E' = \{e'_{ij} = (a_i \in A, v_j \in H)\}$ to find the active node degree g_j of each sensing hole $v_j \in H$ such that $g_j = |\{a_i \in A : d'_{ij} \leq R\}|$ and d'_{ij} stands for the Euclidean distance between $a_i \in A, v_j \in H$.

We want to find a set of b sensor relocation trajectories $\Phi = \{\varphi_k\}$, one for each robot, where $\varphi_k = (v_{\gamma_1}, v_{\gamma_2}, \dots, v_{\gamma_{|\varphi|}})$ where $\gamma : [1; |V| - 1] \rightarrow \{\gamma_i\}, v \notin V$ is an injective function, that starts and ends at the base station. Falcon et al. [4] state that a sensor relocation trajectory is feasible if it has no duplicate nodes (other than the base station being its origin and termination), repairs all sensing holes, and never violates the capacity constraint, Q_{max} . As we are creating trajectories for multiple robots we instead state that Φ must repair all sensing holes, rather than an individual φ_k . We use the following objective functions to evaluate the quality of Φ :

Minimize Trajectory Length

$$\sum_{k=1}^b \sum_{e_{ij} \in \varphi_k} d_{ij} \quad (1)$$

Maximize Trajectory Robustness

$$\sum_{v_i \in \Phi \cap S, v_j \in \Phi \cap H} p_i g_j \delta_{ij} \quad (2)$$

Maximize Trajectory Lifetime

$$\min\{p_i\}, v_i \in \Phi \cap S \quad (3)$$

Minimize Load Balancing

$$\max[(\sum_{e_{ij} \in \varphi_k} d_{ij} / (\sum_{l=1}^b \sum_{e_{ij} \in \varphi_l} d_{ij}))], \varphi_k \in \Phi \quad (4)$$

The trajectory robustness is representative of how likely a sensor hole would be to cause a disconnect in the WSN. It is expressed as a value based on the battery power of a given sensor and its score, g_j . This score is determined using the concept of *ab vertex separation* and the following equation:

$$g_j = 1 + \sum_{a \in A} \sum_{v_j \neq a, v_j \neq v_i, v_j \in C_{abi}} 1/|C_{abi}|, v_j \in A \quad (5)$$

where C_{abi} is the i th ab-separator [19], $a \in A$, and $b = v_0$ the depot.

The function δ_{ij} is evaluated as follows:

$$\delta_{ij} = \begin{cases} 1 & \text{if } v_i \in \varphi \cap S \text{ will be dropped off at} \\ & \text{the location of } v_j \in \varphi \cap H \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

The trajectory lifetime shows the worst passive sensor power level in a sensor relocation trajectory. We want the selected passive sensors to have enough battery power so they can sustain operational demands without being depleted in a short period of time, hence becoming damaged nodes themselves. The load balancing objective exists to try and evenly distribute the total distance travelled between all robots.

IV. EMOO ALGORITHMS FOR RMRASR

This section details the various components of the evolutionary multi-objective optimizers that aim at solving the RMRASR problem.

A. Solution Encoding

We model our solutions as a modified version of the *permutation-based* representation used in [5] and [18]. For a problem instance that contains n nodes we create a permutation of size $(n - 1) + (r - 1)$ where r is the number of robots. The extra $r - 1$ nodes are copies of the depot node, v_0 , used to divide the solution into r distinct sections, one for each of the robots. It is assumed that the beginning and ending of the permutation are v_0 , the base station node. Thus, each solution will have the form $(v_1, \dots, v_{n+r-2})$ to represent the order which the nodes are visited. We indicate the unvisited nodes in the permutation with negative numbers. For example, the permutation $(2, -3, 1, 0, 6, 5, -4)$ would indicate that robot 1 would visit the nodes 2 and 1 and robot 2 would visit 6 and 5.

B. Objective Functions

We use (1), (2), (3), and (4) as the objective functions for our problem. As a result, each solution is evaluated in terms of these four unique (and conflicting) objectives.

C. Population Initialization

The population is randomly initialized by creating a random permutation, with no duplicate elements, and assigning the sign of each element (positive or negative) with equal probability.

D. Evolutionary Operators

1) *Selection Operator*: *Binary tournament* [20] is used as the selection operator for our experiments. It works by randomly selecting two individuals from the population (or archive, depending on the algorithm) and selecting the best. The non-domination rank [21] is used as the comparison criterion, rather than any of the fitness values. If selected solutions share the same non-domination rank, either solution is chosen with equal probability.

2) *Crossover Operator*: We use *Partially-Mapped Crossover* (PMX) [22] as our crossover operator. PMX builds offspring solutions by selecting a subsequence from one of the parents and inserting it into the other, preserving as many points as possible. PMX is designed such that duplicate entries do not form in the permutation.

3) *Mutation Operator*: Our implementation uses two mutation operators in tandem: *swap* and *insert*. Each offspring has a chance to undergo mutation. The *swap* operator works by randomly selecting two elements of the permutation and switching their values. The *insert* operator works by selecting an element of the permutation and inserting it at another random location in the permutation.

Due to the nature of our encoding scheme, these mutations would have little or no effect if there were only to affect the negative elements of a permutation. Therefore, any element that is affected by a mutation will have its sign flipped.

E. Infeasibility Handling

We use a modified version of the repair sequence used in [5] to account for the addition of multiple robots. This repair function is made up of two components: *content repair* and *order repair*.

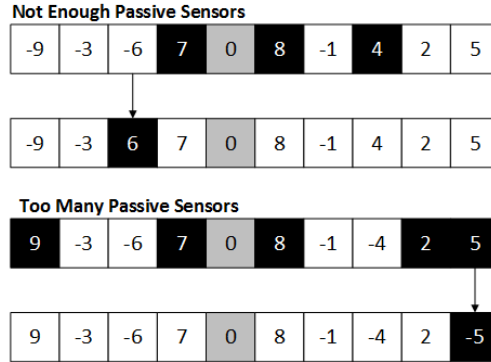


Fig. 1. In the first example there are not enough passive sensors in the relocation trajectory. We add passive sensors from right to left until we have enough. In the second example there are too many passive sensors in the relocation trajectory. We remove passive sensors from left to right.

1) *Content Repair*: If the permutation does not contain all of the damaged sensors, an appropriate number of passive sensors, or all of the duplicate depots then the following procedure is enacted:

- In the case $B \subseteq \Phi$ is not true, we add the remaining elements of B to Φ . We do this by making all elements belonging to B in the encoded solution positive if they are not already.
- In the case $H \subseteq \Phi$ is not true, we add the remaining elements of H to Φ . We do this by making all elements belonging to H in the encoded solution positive if they are not already.
- If $|\Phi| > 2|H| - Q_0$, remove $v_i \in S \cap \Phi$ from right to left until $|\Phi| = 2|H| - Q_0$. Going from right to left prevents us from collecting passive sensors after the last sensing hole is visited. We iterate through the permutation from $n + b$ to 1 and flip the sign of those v_i from positive to negative until the condition is satisfied.
- If $|\Phi| < 2|H| - Q_0$, add $v_i \in S, v_i \notin \Phi$ from left to right until $|\Phi| = 2|H| - Q_0$. Going from left to right prevents us from removing passive sensor collections that are being used as part of the solution. We iterate through the permutation from 1 to $n + b$ and flip the sign of those v_i from negative to positive until the condition is satisfied.

The content repair function is illustrated in Figure 1.

2) *Order Repair*: The ordering constraint dictates that: at all points in Φ , $0 \leq Q \leq Q_{max}$ must hold. In addition, we must also guarantee that each robot completes at least one pickup and delivery, i.e. $Q = 0$ for all $v_i \in \Phi \cap B$. We must consider the infeasible cases for repair:

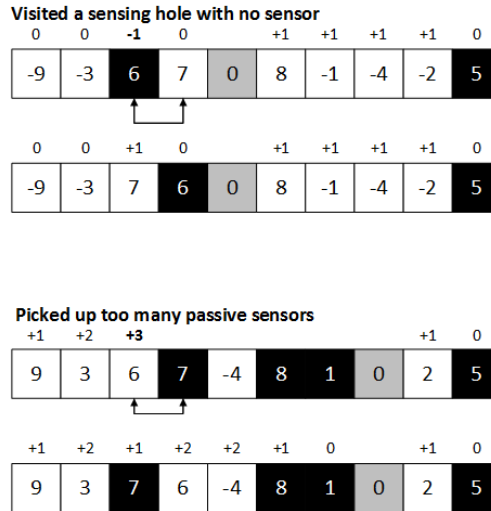


Fig. 2. Examples of sensor relocation trajectory ordering repair for $Q_{max} = 2$. In the first example the robot arrives at a sensing hole without a replacement. We swap the sensing hole with the next passive sensor. In the second example the robot attempts to collect a passive sensor when it already has a full load. We swap the passive sensor with the next sensing hole.

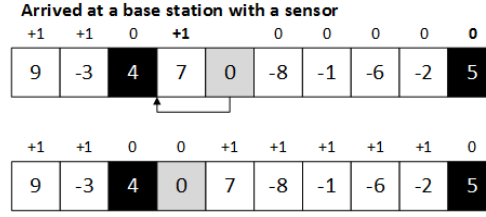


Fig. 3. Example of sub-trajectory validity repair. The robot arrives at the base station while carrying a sensor. We insert the offending base station node before the offending pickup.

- If a trajectory, φ_k , contains no deliveries, then the next sensing hole, $v_i \in H$ in sequence is inserted before the offending $v_j \in B$. If there are no more sensing holes then $v_j \in B$ is inserted at the beginning of the permutation. This process is repeated until all trajectories contain at least one sensing hole.
- If $Q > Q_{max}$ at some point in Φ , the offending $v_i \in \Phi \cap S$ is moved after the nearest right $v_i \in \Phi \cap H$ in Φ . To do this we insert the element in which the issue appears after the rightmost element of H that appears in the permutation, as illustrated in Figure 2.
- If $Q > 0$ at some point in $v_i \in \Phi \cap B$, move the closest, leftmost $v_j \in \Phi \cap S$ after $v_k \in \Phi \cap H$. To do this we remove the pickup that is closest on the left to the offending depot and place it before the next delivery in sequence as illustrated in Figure 3.
- If $Q < 0$ at some point in Φ , move the next $v_j \in \Phi \cap S$ before the offending $v_i \in \Phi \cap H$ in Φ . To do this we move the next pickup in sequence before the offending delivery as illustrated in Figure 2.

F. Stop Criteria

Each algorithm is terminated when it reaches the 200,000 budget in the number of objective function evaluations.

V. EXPERIMENTAL RESULTS

This section details the experimental methodology and evaluation of our proposed MOO solution for RMRASR.

A. Experimental Setup

1) *Synthetic Scenario Generation*: We first generate a WSN by mapping a grid of active sensors to a ROI, then flag a random subset of these sensors as the sensing holes. We complete the scenario by using the *MAX_DPA* algorithm adapted from [23]. The parameters of the scenario, such as: the total number of sensors N , ratio of sensing holes to passive sensors R_{HS} , and distribution density of passive sensors, are determined prior to scenario generation. The resulting scenarios provide the set of files used for experimentation.

In order to have a reference point for our selected performance metrics, we also generated reference sets for each of the instance files. Each algorithm was run 30 times for each file and their final Pareto sets combined to form the reference set for that file.

We generated a total of 99 scenarios¹: 50 scenarios for varying N from 20 to 1000 in increasing steps, and 49 for differing R_{HS} values from 2% to 50%.

2) *Benchmark EMOO Algorithms*: We examine the performance of six state-of-the-art EMOO algorithms on the RMRASR problem. We include SPEA2 [24] which uses a strength function based on the number of dominated solutions of the non-dominated set. PESA-II [25] relies on a region-based approach that divides the solution space into hypercubes; solutions are chosen to keep the best and maintain a healthy spread across hypercubes. NSGA-II [21] and its successor, NSGA-III [26], use the Pareto dominance of solutions as well as a crowding distance operator. NSGA-III follows the same idea as NSGA-II, but also uses reference planes to navigate towards good solutions. AGE-I [27] and AGE-II [28] aim to have the population model approximate the solutions space. AGE-II differs from its predecessor by incorporating ε -dominance [29] and is stated to improve both solution quality and runtime.

In addition to the algorithms mentioned above, we include an application of the hybrid LifeCycle (LC) model [6]. Our implementation of LC uses the three best algorithms, as determined by initial experimentation, as its basis: SPEA2, AGE-I, and NSGA-III. When the performance of one of these algorithms falls below a certain threshold (lack of progress for five consecutive generations) the solution population is passed to the next algorithm in sequence. Our threshold is as follows:

¹http://www.site.uottawa.ca/~rfalc032/files/rmraser_materials.zip

$$\frac{P'}{P} < 0.90 \quad (7)$$

where P is the number of solutions in the previous approximation set and P' is the number of solutions in both the current and approximation sets. This cycle executes until, as normal, all function evaluations have been completed.

3) *Algorithm Parameter Configurations*: All algorithms were run using the following parameter values: population size = 200, number of function evaluations = 200,000, probability of crossover = 0.75, and probability of mutation = 0.25 in order to provide a fair comparison baseline.

4) *Performance Metrics*: We use a set of well-known MOO indicators that judge how good an obtained Pareto approximation set is from different angles, i.e., hypervolume (HV), maximum Pareto front error (MPE), spacing (SPC), generational distance (GD), inverted generational distance (IGD), and runtime (RT). For a description of these metrics refer to [30] and Section 5.4.4 in [31]. The reported values for the metrics are averages, over each synthetic scenario, of the means of 30 independent runs of an algorithm upon a scenario.

5) *Statistical Validation*: To confirm whether or not the differences in observed performance are statistically significant at the 5% level, we rely on nonparametric tests as suggested by [32], where applicable.

In order to comment on algorithm performance we must establish or reject significant differences within the whole group of algorithms using the Friedman $N \times N$ procedure. When a test rejects the null hypothesis then a set of comparisons is done using the following post-hoc procedures: Bergmann, Nemenyi, Holm, and Schaffer [32]. We default to the Bergmann procedure given its robustness and reliable adjustment of p -value. In the case that the Bergmann procedure does not reject the null hypothesis of a comparison, we will look to the other post-hoc methods.

The algorithm performance data used for the statistical validation procedure was collected at 25, 50, 75, and 100 percent of the execution budget across each of our scenarios.

B. Experiment 1: Number of Robots Analysis

We examine how the algorithms behave for a varying number of robots in the problems. We focus on a problem instance with 200 nodes and 5% damaged nodes as it is representative of the experimentation testbed.

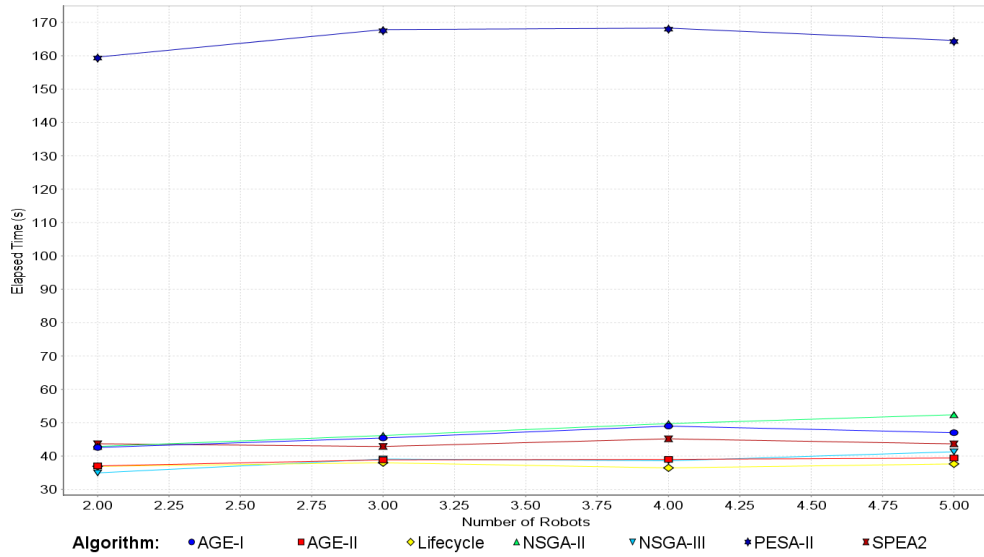


Fig. 4. Average time, over 30 executions, for the problem instance to complete with varying amounts of robots

The execution time of the algorithms remains relatively constant as seen in Figure 4. The repair function did not lead to an increase in the overall runtime despite a higher number of infeasible subtours to repair. Notice that PESA-II takes 3-4 times longer to finish, this is due to the size of the calculations that PESA-II does in order to use the hypercubes that help maintain a healthy spread of solutions. The results found in [5] showed that PESA-II was the fastest performing algorithm, but PESA-II is the only algorithm in which the dimensionality of the problem would drastically increase the amount of calculations required.

While the algorithms should not be affected by the number of robots, the repair function could possibly have caused an increase in overall execution time for all algorithms, which we can see is not the case.



Fig. 5. Normalized values for objective functions (2), (3), and (4) using the Pareto set from the generated reference set for the representative instance file

Figure 5 suggests that the number of robots used to solve a problem does have an impact on the solution quality. By using our reference set generated from multiple runs of all of our algorithms, we believe that we have a representative set of solutions for the problem instance. The objective functions (2), (3), and (4) have calculable minimum and maximum values and therefore can be appropriately normalized, as seen in Figure 5, for comparison. The three-robot configuration appears to show the least amount of variance in solutions for the Trajectory Lifetime and Trajectory Robustness, while the other algorithms seem to perform similarly.

We can see that the normalized Load Balancing values remain centralized around the same point for all numbers of robots examined. This would indicate that the addition, or subtraction, of robots from a system would not detract from the ability to correctly distribute the workload.

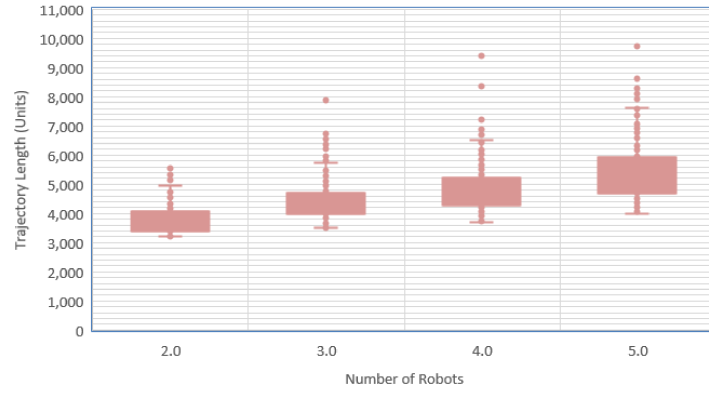


Fig. 6. Values for Trajectory Length (1) using the Pareto set from the generated reference set for the representative instance file

Figure 6 reveals to us that the more robots are involved in a solution the longer the total distance will be. This may be because of the disjoint tours requiring more travel overall due to all robots being deployed from the base station and having to return there. The advantage would be that the total distance travelled is distributed between the robots and therefore the replacement procedure would take less time as the work is being completed in parallel. As seen in Figure 5 the load balancing seems to be done quite well by the algorithms, generally above 80 percent of the optimal value. This experiment shows that our formulation is able to provide good solutions to the RMRASR problem regardless of the number of robots deployed.

C. Experiment 2: Scalability Analysis

In this experiment we examine the effect of WSN size on the RMRASR solutions. The performance metrics of our selected algorithms are evaluated across WSNs ranging from 20 to 1,000 nodes, with the percentage of damaged nodes set to 15. We show the results of our experiments on the generated instance files using 3 robots as this better highlights the differences in algorithm performance.

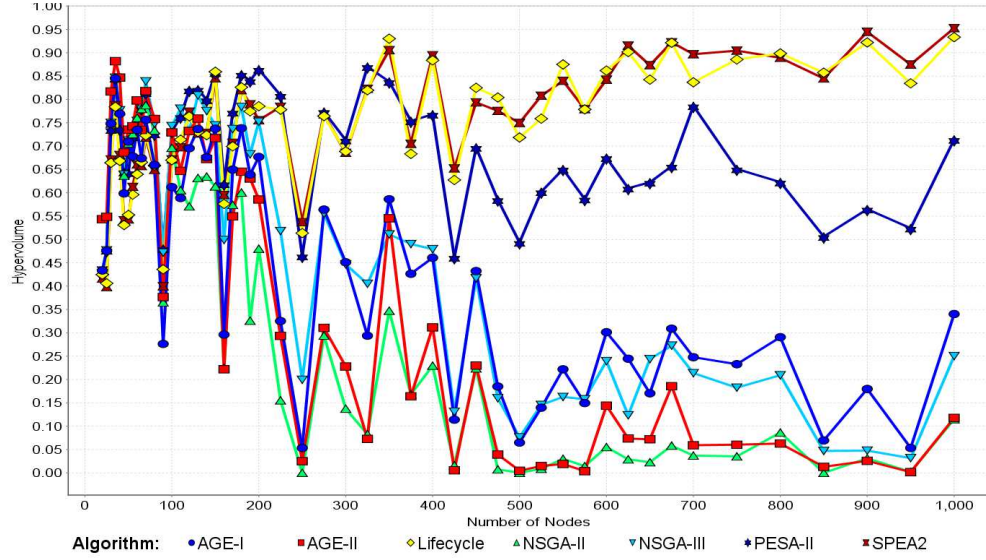


Fig. 7. Final execution results for HV across all generated scenarios for the scalability analysis experiment

Figure 7 shows the performance of each of the algorithms based on the HV metric across all of the experiment instances. We can see that SPEA2 and LC far outperform the other algorithms. The solution strength metric used by SPEA2 clearly performs well given the number of objectives and the large solution space. LC performs similarly well as our implementation is based on SPEA2. We can see that algorithms like NSGA-II and NSGA-III that rely largely on Pareto-dominance do not fare well in this space as the crowding distance mechanism likely does not come into play enough to encourage exploration. PESA-II performs reasonably well, perhaps due to the archive set as it uses *hypercubes* to ensure a good spread of solutions, preventing it from getting caught in local maxima. AGE-I uses approximation to guide its solutions, but the approximation is based on its non-dominated archive set. This causes the solutions to be driven towards the existing archive, which may or may not be good. AGE-II takes the same approach, but uses ε -dominance to try and control the size of the archive in order to reduce runtime. We can see that this causes a performance tradeoff as it behaves worse than AGE-I.

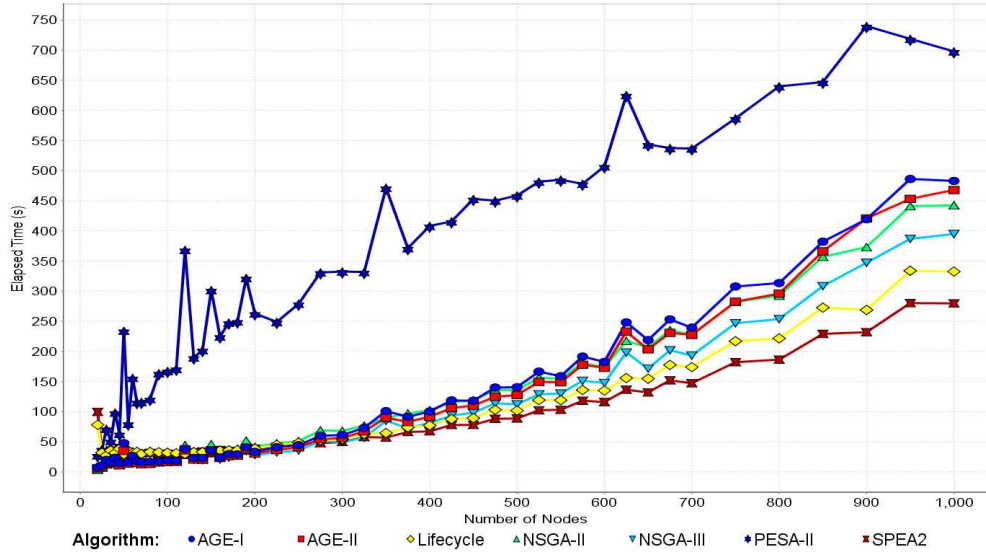


Fig. 8. Final execution results for time elapsed, in seconds, across all generated scenarios for the scalability analysis experiment

We can see in Figure 8 that as the solution space increases in size the amount of runtime has an increasing trend. PESA-II

is not exception to this rule, but has a very poor runtime as compared to the other algorithms. The *hypercubes* that PESA-II uses require a large amount of calculations to maintain; as these calculations must consider four dimensions, one for each objective function, they do not seem to scale well. The other algorithms that are based more on Pareto dominance do not seem to be affected by this. We can also see that AGE-II is slightly faster than AGE-I in most cases, lending credibility to the statements that the ε -dominance archive positively affects the speed of the algorithm. SPEA2 and LC tend to perform the best overall as the strength metric used by SPEA2 is fast to calculate and differentiate among solutions. The exception to this is when the solution space is very small, where there is little to differentiate solutions and the strength calculation algorithm is likely running at close to worst case.

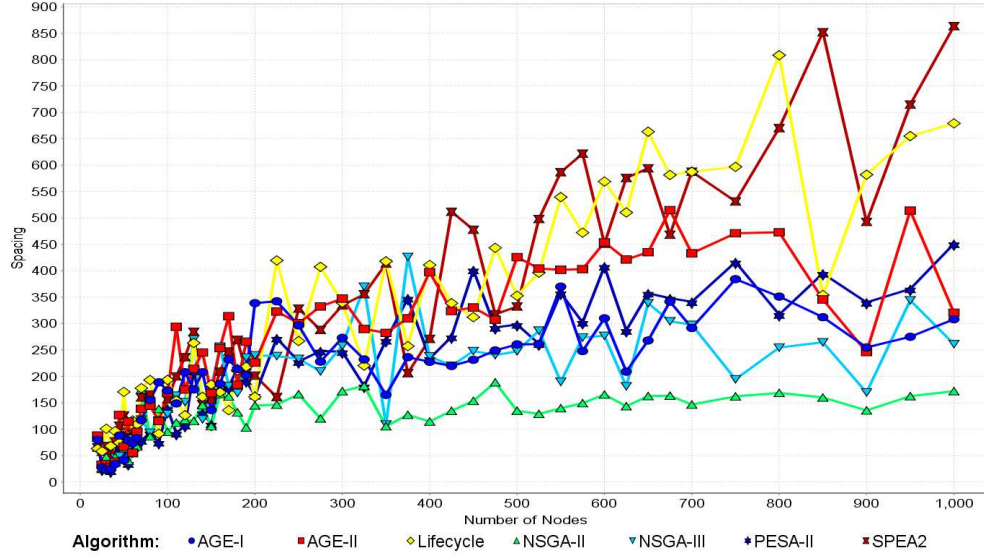


Fig. 9. Final execution results for spacing across all generated scenarios for the scalability analysis experiment

The results of the spacing metric are shown in Figure 9. We can see that as the number of nodes increases, along with the size of the solution space, this metric remains fairly constant, with the exception of SPEA2, LC, and AGE-II. As the size of the solution space grows along with the number of solutions that dominate and can be dominated, it should be expected that the spacing of each algorithm would increase. This is important to note as there is a very large number of possible solutions at each domination rank due to the high dimensionality of the problem. NSGA-II and NSGA-III do not have any explicit mechanism for encouraging the exploration of a large solution space. AGE-II is an improvement on AGE-I in this instance, likely due to the selective nature of the ε -dominance archive. PESA-II uses hypercubes to try and guarantee diverse solutions, but the number of hypercubes remained static across all problems and was therefore unable to adapt to the growing size of the solution space. SPEA2, and by association LC, employ a strength function, which considers the solution proximity, to assign weights to its solutions. The benefits of this approach can be seen in Figure 9.

Figure 10 shows algorithm improvement, in terms of the MPE, over the course of execution. NSGA-II is a standout in terms of its poor performance potentially due to the crowding distance function not coming into effect as the large solution space provides many ranks of solutions. The dimensionality of our problem is quite large, and as a result there are many local optima. NSGA-II likely gets trapped in these as it uses only non-domination for selecting its solutions, causing the performance reported in Figure 10.

TABLE II
EXPERIMENT 2 - FRIEDMAN $N \times N$ TEST RESULTS

Metric	Algorithm Rank							<i>p</i> -value
	NSGA-II	NSGA-III	SPEA2	PESA-II	AGE-I	AGE-II	LC	
HV	5.735	4.01	2.5725	2.6175	4.7875	5.0125	3.265	1.4117E-10
SPC	6.3175	5.0025	2.37	4.32	4.51	3.235	2.245	2.1050E-10
MPE	6.725	4.795	1.915	2.975	4.815	4.575	2.2	2.3401E-10
GD	5.5225	3.3225	3.655	1.725	4.05	5.47	4.255	1.5108E-10
IGD	5.1825	3.9775	3.29	2.16	4.41	5.395	3.585	1.6803E-10
RT	4.235	2.385	2.755	6.97	4.75	3.245	3.66	2.0868E-10

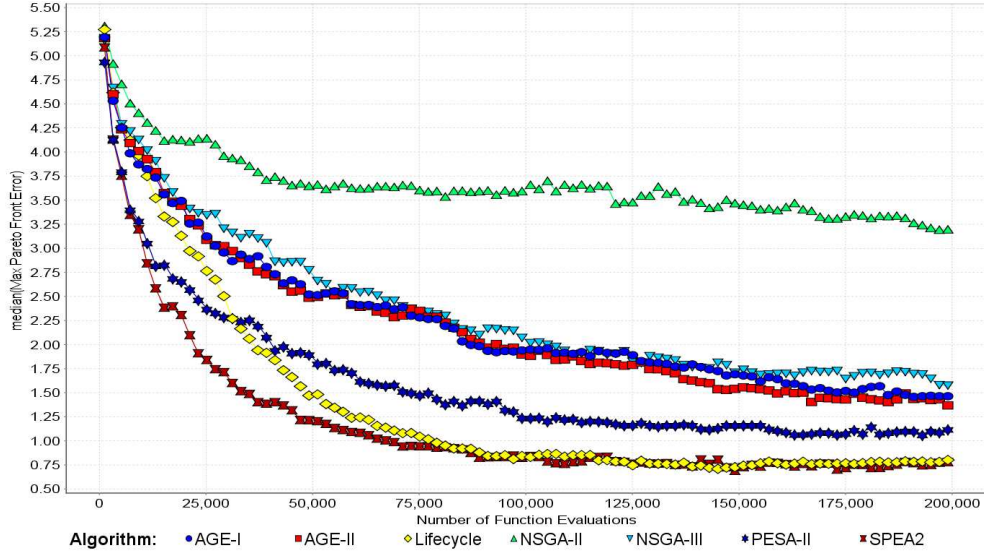


Fig. 10. Median results across algorithm execution for maximum Pareto front error for generated scenarios for the scalability analysis experiment

Table II gives an overview of the statistical analysis carried out for each of the recorded metrics. The small p -values indicate that the results of the analysis are statistically significant for the group of algorithms under each metric. For each metric the following null hypothesis are not rejected:

HV: AGE-I vs. AGE-II, SPEA2 vs. PESA-II

SPC: PESA-II vs. AGE-I, SPEA2 vs. LC

MPE: NSGA-III vs. AGE-I, NSGA-III vs. AGE-II, AGE-I vs. AGE-II, SPEA2 vs. LC

GD: NSGA-II vs. AGE-II, AGE-I vs. LC, NSGA-III vs. SPEA2, SPEA2 vs. AGE-I

IGD: NSGA-II vs. AGE-II, SPEA2 vs. LC, NSGA-III vs. LC, NSGA-III vs. AGE-I

RT: NSGA-III vs. SPEA2, AGE-II vs. LC, SPEA2 vs. AGE-II

The results of our statistical analysis, including post-hoc procedures, show that our results are generally statistically significant in terms of *post-hoc* comparisons. It is also interesting to note that while PESA-II does perform quite well, we can see in Figure 8 and Table II that it is by far the slowest algorithm. Also interesting to note are the cases where SPEA2 vs. LC is not a rejected null hypothesis; this is probably due to the fact that SPEA2 is the first component of our LC implementation. Overall it seems that SPEA2 would be a better selection than LC, as the addition of the other algorithms into LC seems to cause poorer performance than SPEA2. Due to the small number of required calculations, NSGA-III performs better in terms of runtime, though the NSGA-III vs. SPEA2 null hypothesis is not rejected for this metric. Examining the HV and MPE metrics we can see that there is no statistically significant difference between AGE-I and AGE-II; this is interesting as AGE-II is marketed as an improvement over AGE-I.

D. Experiment 3: Inflicted Damage Analysis

In this experiment we examine the effect of various WSN damage levels on the RMRASR solutions. The performance metrics of our selected algorithms are evaluated across WSNs with 2 to 49 percent damaged sensors, with the number of nodes set to 200. To maintain consistency, we again used 3 robots.

Figure 11 illustrates HV values across various levels of inflicted damage to the WSN. We can see that PESA-II, SPEA2, and LC tend to perform consistently across all levels of inflicted damage, whereas the other algorithms generally seem to decrease in performance as the level of inflicted damage increases. This is likely because SPEA2, and by association LC, as well as PESA-II are designed to use dominant, distinctive solutions as the bases for reproduction. This will encourage enough exploration that these algorithms are able to exploit the solution space and avoid local optima. This is especially important in our problem due to the dimensionality of the solution space. We can also see that below 10 percent WSN damage the best performing algorithms, PESA-II and SPEA2, do not fare decently. This would be a point worth considering when deciding on an algorithm for real-world implementation.

Figure 12 reveals that RT across all levels of inflicted damage is relatively constant, with the exception of PESA-II which is grossly outperformed by all of the other algorithms under consideration. Due to the stochastic nature of our instance file

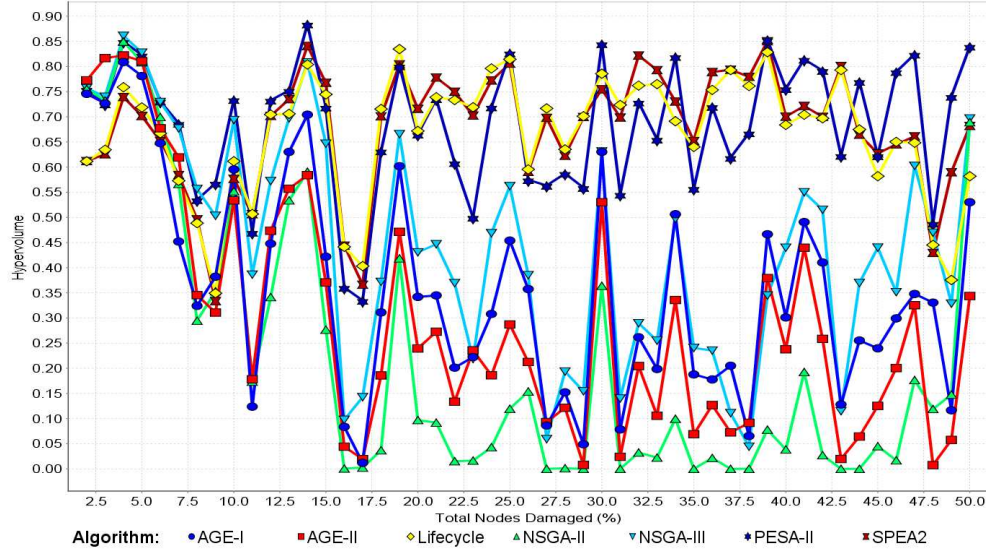


Fig. 11. Final execution results for HV across all generated scenarios for the inflicted damage experiment

TABLE III
EXPERIMENT 3 - FRIEDMAN $N \times N$ TEST RESULTS

Metric	Algorithm Rank							p -value
	NSGA-II	NSGA-III	SPEA2	PESA-II	AGE-I	AGE-II	LC	
HV	6.0842	4.051	2.0918	2.2117	5.1122	5.4923	2.9566	2.6346E-10
SPC	6.6224	4.301	3.4082	4.0408	4.1122	2.5102	3.0051	2.0486E-10
MPE	6.8367	5.0357	1.6173	2.8469	4.7041	4.8622	2.0969	2.4429E-10
GD	5.648	3.352	3.1378	1.5357	4.25	6.0663	4.0102	1.9649E-10
IGD	6.2398	4.0867	2.3316	2.0459	4.7449	5.5204	3.0306	2.2263E-10
RT	5.3163	2.1735	2.9949	7	4.4949	2.8316	3.1888	2.2168E-10

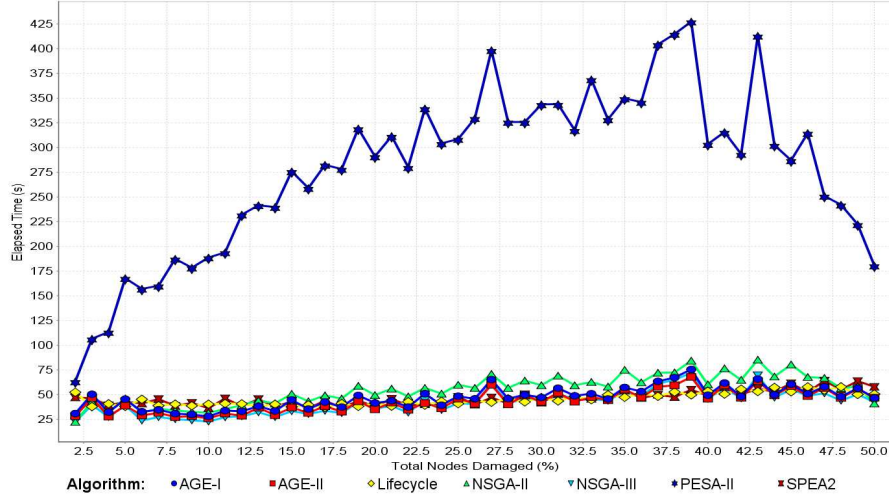


Fig. 12. Final execution results for RT, in seconds, across all generated scenarios for the scalability analysis experiment

generation, it is difficult to pinpoint the cause of the fluctuation in the execution time of PESA-II; however, as mentioned in Section V-B, the dimensionality of the problem is the likely cause of the poor performance of PESA-II compared to the other algorithms since the repair and evaluation functions are algorithm agnostic.

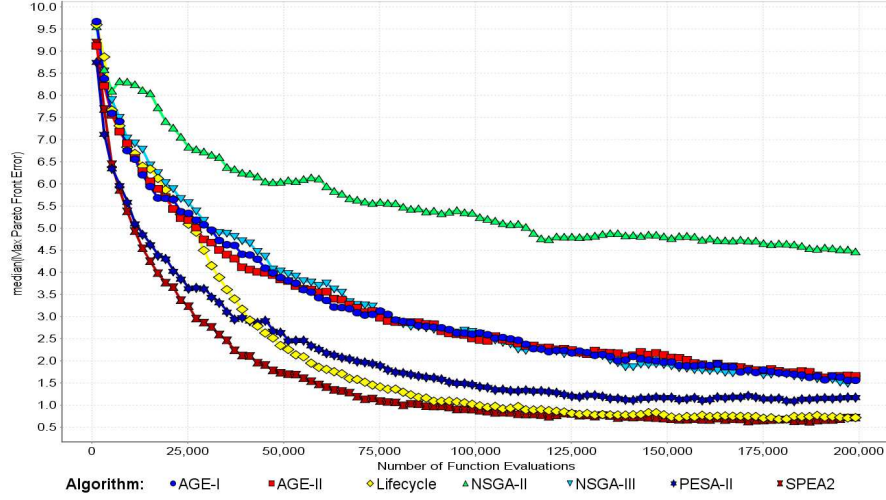


Fig. 13. Median results across algorithm execution for MPE for generated scenarios for the inflicted damage experiment

Figure 13 shows the MPE over the course of algorithm execution. We can see, much like Figure 10, that NSGA-II performs significantly worse than the other algorithms in this regard, likely due to getting trapped in a local optima with solutions being far from the reference set. The algorithms appear to improve the most between 0 and 50,000 objective function evaluations, which is a quarter of the execution allotment. The three best performing algorithms, SPEA2, LC, and PESA-II, exhibit little improvement after 100,000 executions. This would indicate that in a real-world setting the number of executions could be reduced in order to shorten runtime with minimal detriment to the quality of the final solutions.

Table III gives an overview of the statistical analysis carried out for each of the recorded metrics. The small p -values indicate that the results of the analysis are statistically significant for the group of algorithms under each metric. For each metric the following null hypothesis are not rejected:

HV: SPEA2 vs. PESA-II, AGE-I vs. AGE-II

SPC: PESA-II vs. AGE-I, NSGA-III vs. AGE-I, NSGA-III vs. PESA-II, SPEA2 vs. LC, AGE-II vs. LC

MPE: AGE-I vs. AGE-II, NSGA-III vs. AGE-II, NSGA-III vs. AGE-I, SPEA2 vs. LC

GD: NSGA-III vs. SPEA2, AGE-I vs. LC, NSGA-II vs. AGE-II

IGD: SPEA2 vs. PESA-II

RT: SPEA2 vs. AGE-II, SPEA2 vs. LC, AGE-II vs. LC

The results in Table III show us that PESA-II generates good solutions according to our chosen metrics, compared to the other algorithms. Even in the case of HV, where SPEA2 is the best, we can see from the Bergmann procedure that there is no significant difference between PESA-II and SPEA2. The issue with PESA-II lies in how long it takes to run, which is also illustrated in Figure 12. We can also see that AGE-I and AGE-II both have their advantages, but give poorer results than the best-performing algorithms. The exception to this is that AGE-II gives a more diverse spread of solutions, though the difference between it and LC is not statistically significant.

VI. CONCLUSIONS

We extended the RRASR problem [5] to account for the multiple-robot case, which we called RMRASR. We used the presence of multiple robots to include a load balancing objective which was shown to give well-balanced trajectories for all

TABLE IV
OVERALL EFFICIENCY AND EFFICACY

		Small	Medium	Large
	Num. Nodes	≤ 150	$> 150, < 500$	≥ 500
Exp. 2	Efficiency	AGE-II	NSGA-III	SPEA2
	Efficacy	AGE-II	SPEA2	SPEA2
	% Damaged	≤ 15	$> 15, \leq 30$	> 30
Exp. 3	Efficiency	NSGA-III	NSGA-III	NSGA-III
	Efficacy	NSGA-III	AGE-II	NSGA-III

of the robots. Additionally, we tackled this problem using six EMOO algorithms using a standard set of genetic operators and included an implementation of the LC hybrid method. We see in our results that SPEA2 and PESA-II are the best performing algorithms, though the addition of a fourth objective function seems to have caused a large runtime increase for PESA-II. We also found that LC was outperformed by one of its constituent algorithms, SPEA2. Interestingly, by examining the algorithms in terms of efficiency (RT) and efficacy (HV, SPC, and MPE), in Table IV, we show that AGE-II, NSGA-III, and SPEA2 are the overall best algorithms for varying problem scenarios. In the future, we would like to explore the efficacy of dedicated combinatorial MOO algorithms as well as an implementation containing some form of local search.

REFERENCES

- [1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.
- [2] N. Mitton and D. Simplot-Ryl, *Wireless Sensor And Robot Networks From Topology Control to Communication Aspects*. WorldScientific, 2014.
- [3] R. Falcon, *Towards Fault Reactiveness in Wireless Sensor Networks with Mobile Carrier Robots*. PhD thesis, University of Ottawa, Ottawa, Ontario, Canada, 2012.
- [4] 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 *Proc. of the IEEE Congress on Evolutionary Computation (CEC)*, (Barcelona, Spain), pp. 4326–4333, 2010.
- [5] B. Desjardins, R. Falcon, R. Abielmona, and E. Petriu, "A multi-objective optimization approach to reliable robot-assisted sensor relocation," in *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pp. 956–964, IEEE, 2015.
- [6] T. Krink and M. Løvbjerg, "The lifecycle model," 2002.
- [7] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 31, no. 12, pp. 1985–2002, 2004.
- [8] H. Min, "The multiple vehicle routing problem with simultaneous delivery and pick-up points," *Transportation Research Part A: General*, vol. 23, no. 5, pp. 377–386, 1989.
- [9] B. P. Bruck and A. G. dos Santos, "Hybrid approach for the multiple vehicle routing problem with deliveries and selective pickups," in *Hybrid Intelligent Systems (HIS), 2012 12th International Conference on*, pp. 265–270, IEEE, 2012.
- [10] J. Berger and M. Barkaoui, "A new hybrid genetic algorithm for the capacitated vehicle routing problem," *Journal of the operational Research Society*, vol. 54, no. 12, pp. 1254–1262, 2003.
- [11] T. J. Ai and V. Kachitvichyanukul, "Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem," *Computers & Industrial Engineering*, vol. 56, no. 1, pp. 380–387, 2009.
- [12] B. M. Baker and M. Ayechew, "A genetic algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 30, no. 5, pp. 787–800, 2003.
- [13] R. Nallusamy, K. Duraiswamy, R. Dhanalaksmi, and P. Parthiban, "Optimization of multiple vehicle routing problems using approximation algorithms," *arXiv preprint arXiv:1001.4197*, 2010.
- [14] Y.-H. Huang and C.-K. Ting, "Genetic algorithm with path relinking for the multi-vehicle selective pickup and delivery problem," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pp. 1818–1825, June 2011.
- [15] B. Ombuki, B. J. Ross, and F. Hanshar, "Multi-objective genetic algorithms for vehicle routing problem with time windows," *Applied Intelligence*, vol. 24, no. 1, pp. 17–30, 2006.
- [16] H. Li, A. Barnawi, I. Stojmenovic, and C. Wang, "Market-based sensor relocation by robot team in wireless sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 22, pp. 259–280, 2014.
- [17] R. Falcon, X. Li, A. Nayak, and I. Stojmenovic, "A Harmony-Seeking Firefly Swarm to the Periodic Replacement of Damaged Sensors by a Team of Mobile Robots," in *2012 IEEE International Conference on Communications (ICC)*, (Ottawa, Canada), pp. 6436–6440, June 2012.
- [18] X.-L. Liao and C.-K. Ting, "An evolutionary approach for the selective pickup and delivery problem," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pp. 1–8, IEEE, 2010.
- [19] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*. Radarweg 29, 1043 NX Amsterdam, Netherlands: Elsevier, 2 ed., 2 2004.
- [20] B. L. Miller and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex Systems*, vol. 9, no. 3, pp. 193–212, 1995.
- [21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, Apr. 2002.
- [22] D. E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," in *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, vol. 154, Lawrence Erlbaum, Hillsdale, NJ, 1985.
- [23] F. A. Onat, I. Stojmenovic, and H. Yanikomeroglu, "Generating random graphs for the simulation of wireless ad hoc, actuator, sensor, and internet networks," *Pervasive and Mobile Computing*, vol. 4, no. 5, pp. 597–615, 2008.
- [24] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," 2001.
- [25] D. W. Corne, N. R. Jerram, J. D. Knowles, M. J. Oates, et al., "PESA-II: Region-based selection in evolutionary multiobjective optimization," in *Proc. of the Genetic and Evolutionary Computation Conference (GECCO2001)*, pp. 283–290, Morgan Kaufmann Publishers, 2001.
- [26] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *Evolutionary Computation, IEEE Transactions on*, vol. 18, pp. 577–601, Aug 2014.
- [27] K. Bringmann, T. Friedrich, F. Neumann, and M. Wagner, "Approximation-guided evolutionary multi-objective optimization," in *IJCAI Proc.-International Joint Conference on Artificial Intelligence*, vol. 22, p. 1198, 2011.
- [28] M. Wagner and F. Neumann, "A fast approximation-guided evolutionary multi-objective algorithm," in *Proc. of the 15th annual conference on Genetic and evolutionary computation*, pp. 687–694, ACM, 2013.
- [29] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, "Combining convergence and diversity in evolutionary multiobjective optimization," *Evolutionary computation*, vol. 10, no. 3, pp. 263–282, 2002.
- [30] M. R. Sierra and C. A. C. Coello, "Improving pso-based multi-objective optimization using crowding, mutation and-dominance," in *Evolutionary multi-criterion optimization*, pp. 505–519, Springer, 2005.
- [31] C. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007.
- [32] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

Conclusion

This chapter summarizes the thesis by discussing the overall results of the collected works. We also propose future research directions that may develop from the research within the works collected here.

6.1 Discussion

The works contained within this thesis proposed a multi-objective formulation of the *traveling salesman problem with selective pickup and deliveries* [32] with particular application to the WSN domain; this problem was referred to as RRASR. The multi-objective formulation was accomplished by considering real-world factors that arise as part of a possible application of the original optimization problem. By examining the importance of selecting passive sensors that have a high battery level, we implemented the *trajectory robustness* and *trajectory lifetime* objectives. The incorporation of these objectives allows a user trying to solve the RRASR problem to be presented with a set of possible solutions based on the execution of a population-based multi-objective optimization al-

gorithm. Having a set of solutions allows the user to select a sensor relocation trajectory based on their needs at the time without having to modify any problem parameters. The user must be presented with a selection of solutions that are not only good, but also small enough in number and varied enough in objective function values that they are not overwhelmed with choice, but rather presented with a good selection of alternatives describing a suitable tradeoff among all decision objectives under consideration.

Designing objective functions that are representative of the problem and with each providing unique value to the solutions was an important part of the works contained within. In the original RRASR proposal, the *trajectory robustness* objective function was based on an approach that uses the number of neighbours of a sensor to determine its importance within the network, a lower number of neighbouring sensors indicates a larger importance [1]. The issue later discovered with this formulation is that there are some edge cases where nodes are deemed to be much more important to the overall connectivity of the network than they actually are. One specific example occurred with nodes that were on the perimeter of the WSN. These nodes will have a small number of neighbours due to being physically located along the perimeter of the network, but will not have an important role in passing information along to the base station as they will be low-traffic nodes within the communication network. In order to remedy this situation, a new approach was introduced in [2] where we used the *a-b separation* [80] to determine the importance of sensors within the communication network. This provided an objective function that was more representative of the issue that we were trying to address as we were able to assign a score to each sensing hole based on how many communication paths were taken through it. Finally, in the multi-robot version of RRASR, RMRASR, we added a load balancing objective in order to make the population gravitate to solutions wherein each robot would be required to travel roughly the same distance [3]. We used an additional objective function instead of a local search style approach in order to keep the computation costs required by the algorithm at every iteration to a minimum as real-world applications are likely to be time-sensitive.

In order to evaluate our proposed problem formulation, we turned to evolutionary multi-objective optimization algorithms. These were selected due to their comparability

with permutation-based solution representations as well as because they are population-based schemes. We wanted to be able to present a user with a set of solutions so that the solution most appropriate for their operational requirements may be selected. This requires that all solutions in this set be valid solutions. While this may seem unintuitive because the user will only be presented with the set of non-dominated solutions we can not take the chance that the user will need to handle invalid solutions. As such, a solution repair function was introduced that guarantees that each solution will be valid at every iteration. The repair function evolved from what is seen in [1] and [2] to account for the multiple-robot case in [3]. This change modified the repair function so that each robot in the solution was guaranteed to be useful, i.e. completes at least one pickup and delivers it.

Lastly, we empirically evaluated various state-of-the-art algorithms using the RRASR and RMRASR problems. Often algorithms are proposed and evaluated against a set of benchmark problems [77] and those are used as the measure for their effectiveness. Both RRASR and RMRASR have not been used as benchmark problems in the past, therefore we are able to examine the selected algorithms outside of traditional benchmarking problems. We found in [1] that PESA-II [19] was the generally best-performing algorithm. AGE-I [21] and AGE-II [22] claimed to perform better than NSGA-II [14], but our results in [2] showed otherwise. We also found that the addition of a fourth objective, load balancing, in [3] caused PESA-II to no longer be the best-performing algorithm while SPEA2 [20] and NSGA-III [18] generally outperformed the others. We found that our LifeCycle [79] implementation did not have the intended effect. Instead of showing better performance than the sum of its parts it instead showed poorer performance than one of its constituent algorithms, SPEA2. This is likely due to the fact that SPEA2 was the best performing algorithm and having LifeCycle switch from it to other, poorer performing, algorithms was detrimental to the overall result. In the original implementation, there were more types of algorithm used rather than just GAs which may have contributed to this result. These results show an unbiased, non-benchmark, statistically validated empirical look at the performance of these algorithms that, to our knowledge, has not been done elsewhere.

6.2 Future Research Directions

A logical next step for our work would be to implement this system with real hardware and test the real-world efficacy of our proposed solution using a subject matter expert to provide empirical validation. A few avenues for this would be monitoring of a naturally hostile environment, such as nuclear exclusion zones or deep ocean, or large geographic areas that are difficult to maintain, such as farms or orchards. Another avenue is to explore an implementation of a multiple base station version of the problem. This would provide its own unique challenges with regards to routing the robots, especially when looking at a network that must undergo multiple repairs. There are also opportunities to examine the effectiveness of algorithms designed for combinatorial optimization problems, such as: multi-objective genetic local search (MOGLS) [81], Two-Phase Pareto Local Search (2PPLS) [82], or Pareto Local Search (PLS) [83], versus the set of standard algorithms that were evaluated in the works collected here. The algorithms used were selected for benchmarking due to their availability in the widely popular MOEA framework³ and JMetal library⁴ [84]. It would also be interesting to see how applying a fitness penalty and removing the repair function affected the results of the various algorithms. Alternatively, the repair function could be augmented to provide specific operators such as k-opt [69], or exchange moves [85]. The works collected within provide ample opportunity and direction for potential research endeavours.

³<http://moeaframework.org/index.html>

⁴<http://jmetal.github.io/jMetal/>

References

- [1] Desjardins, B. *et al.* *A multi-objective optimization approach to reliable Robot-Assisted Sensor Relocation.* In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 956–964. IEEE (2015). (Cited on pages v, 6, 27, 44, 77, 95, and 96.)
- [2] Desjardins, B. *et al.* *Planning Robust Sensor Relocation Trajectories for a Mobile Robot with Evolutionary Multi-Objective Optimization.* In *Computational Intelligence in Wireless Sensor Networks: Recent Advances and Future Challenges.* Springer (2016). To appear. (Cited on pages v, 6, 43, 77, 95, and 96.)
- [3] Desjardins, B. *et al.* *Reliable Multiple Robot-Assisted Sensor Relocation using Multi-Objective Optimization.* In *Evolutionary Computation (CEC), 2016 IEEE Congress on.* IEEE (2016). To appear. (Cited on pages v, 7, 77, 95, and 96.)
- [4] Yick, J. *et al.* *Wireless sensor network survey.* Computer networks, **52**(12):2292–2330 (2008). (Cited on pages 2, 4, and 8.)

- [5] Simon, G. *et al.* *Sensor network-based countersniper system*. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 1–12. ACM (2004). (Cited on page 2.)
- [6] Chipara, O. *et al.* *Reliable clinical monitoring using wireless sensor networks: experiences in a step-down hospital unit*. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 155–168. ACM (2010). (Cited on page 2.)
- [7] Garg, M. *et al.* *A Novel Optimization Based Energy Efficient Routing Protocol to Increase the Survivability of WSN (wireless sensor network)*. *International Journal of Advanced Research in Computer Science*, **7**(2) (2016). (Cited on page 2.)
- [8] Shi, W. and Corriveau, J.P. *A Comprehensive Review of Sensor Relocation*. In *In Proceedings of the 2010 IEEE/ACM Int’l Conference on Green Computing and Communications & Int’l Conference on Cyber, Physical and Social Computing (GREENCOM-CPSCOM ’10)*, pages 780–785 (2010). (Cited on page 2.)
- [9] Falcon, R. *Towards Fault Reactiveness in Wireless Sensor Networks with Mobile Carrier Robots*. Ph.D. thesis, University of Ottawa, Ottawa, Ontario, Canada (2012). (Cited on page 2.)
- [10] Dunbabin, M. *et al.* *Data muling over underwater wireless sensor networks using an autonomous underwater vehicle*. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2091–2098. IEEE (2006). (Cited on page 2.)
- [11] Yoo, Y. and Agrawal, D.P. *Mobile sensor relocation to prolong the lifetime of wireless sensor networks*. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 193–197. IEEE (2008). (Cited on page 2.)
- [12] Falcon, R. *et al.* *The One-Commodity Traveling Salesman Problem with Selective Pickup and Delivery: an Ant Colony Approach*. In *Proceedings of the IEEE Congress*

- on *Evolutionary Computation (CEC)*, pages 4326–4333. Barcelona, Spain (2010). (Cited on pages 2, 3, and 5.)
- [13] Laporte, G. *The traveling salesman problem: An overview of exact and approximate algorithms*. European Journal of Operational Research, **59**(2):231–247 (1992). (Cited on pages 4, 12, and 14.)
- [14] Deb, K. *et al.* *A fast and elitist multiobjective genetic algorithm: NSGA-II*. IEEE Transactions on Evolutionary Computation, **6**(2):182–197 (2002). (Cited on pages 4, 6, 19, 27, and 96.)
- [15] Balke, W.T. *et al.* *Restricting skyline sizes using weak Pareto dominance*. Informatik-Forschung und Entwicklung, **21**(3-4):165–178 (2007). (Cited on page 5.)
- [16] Liao, X.L. and Ting, C.K. *An evolutionary approach for the selective pickup and delivery problem*. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE (2010). (Cited on page 5.)
- [17] Falcon, R. *et al.* *An evolving risk management framework for wireless sensor networks*. In *Computational Intelligence for Measurement Systems and Applications (CIMSAS), 2011 IEEE International Conference on*, pages 1–6. IEEE (2011). (Cited on pages 6 and 25.)
- [18] Deb, K. and Jain, H. *An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints*. Evolutionary Computation, IEEE Transactions on, **18**(4):577–601 (2014). ISSN 1089-778X. <http://dx.doi.org/10.1109/TEVC.2013.2281535>. (Cited on pages 6, 27, and 96.)
- [19] Corne, D.W. *et al.* *PESA-II: Region-based selection in evolutionary multiobjective optimization*. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 283–290. Morgan Kaufmann Publishers (2001). (Cited on pages 6, 27, and 96.)

- [20] Zitzler, E. *et al.* *SPEA2: Improving the strength Pareto evolutionary algorithm* (2001). (Cited on pages 6, 19, 27, and 96.)
- [21] Bringmann, K. *et al.* *Approximation-guided evolutionary multi-objective optimization*. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1198 (2011). (Cited on pages 6, 44, and 96.)
- [22] Wagner, M. and Neumann, F. *A fast approximation-guided evolutionary multi-objective algorithm*. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 687–694. ACM (2013). (Cited on pages 6, 44, and 96.)
- [23] Li, X. *et al.* *Servicing wireless sensor networks by mobile robots*. *Communications Magazine, IEEE*, **50**(7):147–154 (2012). (Cited on page 9.)
- [24] McCausland, J. *et al.* *Auction-based node selection of optimal and concurrent responses for a risk-aware robotic sensor network*. In *Robotic and Sensors Environments (ROSE), 2013 IEEE International Symposium on*, pages 136–141. IEEE (2013). (Cited on page 9.)
- [25] Pompili, D. *et al.* *Deployment analysis in underwater acoustic wireless sensor networks*. In *Proceedings of the 1st ACM international workshop on Underwater networks*, pages 48–55. ACM (2006). (Cited on page 9.)
- [26] Toupis, S. and Tassiulas, L. *Optimal deployment of large wireless sensor networks*. *Information Theory, IEEE Transactions on*, **52**(7):2935–2953 (2006). (Cited on page 9.)
- [27] Yick, J. *et al.* *Placement of network services in a sensor network*. *International Journal of Wireless and Mobile Computing*, **1**(2):101–112 (2006). (Cited on pages 9 and 10.)
- [28] Batalin, M.A. and Sukhatme, G.S. *The analysis of an efficient algorithm for robot coverage and exploration based on sensor network deployment*. In *Robotics and Au-*

- tation, 2005. *ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3478–3485. IEEE (2005). (Cited on page 10.)
- [29] Li, X. *et al.* *Placing sensors for area coverage in a complex environment by a team of robots*. ACM Transactions on Sensor Networks (TOSN), **11**(1):3 (2014). (Cited on page 10.)
- [30] Falcon, R. *et al.* *A harmony-seeking firefly swarm to the periodic replacement of damaged sensors by a team of mobile robots*. In *Communications (ICC), 2012 IEEE International Conference on*, pages 4914–4918. IEEE (2012). (Cited on pages 10, 11, and 17.)
- [31] Rezazadeh, J. *Mobile wireless sensor networks overview*. International Journal of Computer Communications and Networks (IJCCN), **2**(1) (2012). (Cited on page 10.)
- [32] Falcon, R. *et al.* *The one-commodity traveling salesman problem with selective pickup and delivery: An ant colony approach*. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE (2010). (Cited on pages 10, 11, 15, 16, and 94.)
- [33] Miao, Y. and Yu-Ping, W. *Coverage repair strategies for wireless sensor networks using mobile actor based on evolutionary computing*. Bulletin of Electrical Engineering and Informatics, **3**(3):213–222 (2014). (Cited on page 11.)
- [34] Li, H. *et al.* *Market-Based Sensor Relocation by Robot Team in Wireless Sensor Networks*. Adhoc & Sensor Wireless Networks, **22**(3) (2014). (Cited on page 11.)
- [35] Magklara, K. *et al.* *Node discovery and replacement using mobile robot*. In *Ad Hoc Networks*, pages 59–71. Springer (2012). (Cited on page 11.)
- [36] Fletcher, G. *et al.* *Randomized Robot-Assisted Relocation of Sensors for Coverage Repair in Wireless Sensor Networks*. In *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*, pages 1–5 (2010). ISSN 1090-3038. <http://dx.doi.org/10.1109/VETECF.2010.5594513>. (Cited on page 11.)
- [37] Pisinger, D. *The quadratic knapsack problem—a survey*. Discrete applied mathematics, **155**(5):623–648 (2007). (Cited on page 12.)

- [38] Eiselt, H.A. *et al.* *Arc routing problems, part I: The Chinese postman problem.* Operations Research, **43**(2):231–242 (1995). (Cited on page 12.)
- [39] Laporte, G. *The vehicle routing problem: An overview of exact and approximate algorithms.* European Journal of Operational Research, **59**(3):345–358 (1992). (Cited on pages 12 and 16.)
- [40] Abdul-Razaq, T. *et al.* *A survey of algorithms for the single machine total weighted tardiness scheduling problem.* Discrete Applied Mathematics, **26**(2):235–253 (1990). (Cited on page 12.)
- [41] Morton, A.B. and Mareels, I.M. *An efficient brute-force solution to the network reconfiguration problem.* Power Delivery, IEEE Transactions on, **15**(3):996–1000 (2000). (Cited on page 13.)
- [42] Guan, Z.H. *et al.* *Chaos-based image encryption algorithm.* Physics Letters A, **346**(1):153–157 (2005). (Cited on page 13.)
- [43] Woeginger, G.J. *Exact algorithms for NP-hard problems: A survey.* In *Combinatorial Optimization—Eureka, You Shrink!*, pages 185–207. Springer (2003). (Cited on page 13.)
- [44] Goldreich, O. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press (2008). (Cited on page 13.)
- [45] Courtois, N.T. and Bard, G.V. *Algebraic cryptanalysis of the data encryption standard.* In *Cryptography and Coding*, pages 152–169. Springer (2007). (Cited on page 14.)
- [46] Nagy, G. and Salhi, S. *Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries.* European journal of operational research, **162**(1):126–141 (2005). (Cited on page 14.)
- [47] Christofides, N. *Worst-case analysis of a new heuristic for the travelling salesman problem.* Technical report, DTIC Document (1976). (Cited on page 14.)

- [48] Matai, R. *et al.* *Traveling salesman problem: An overview of applications, formulations, and solution approaches*. INTECH Open Access Publisher (2010). (Cited on page 15.)
- [49] Grötschel, M. and Holland, O. *Solution of large-scale symmetric travelling salesman problems*. Mathematical Programming, **51**(1-3):141–202 (1991). (Cited on page 15.)
- [50] Furini, F. and Malaguti, E. *Models for the two-dimensional two-stage cutting stock problem with multiple stock size*. Computers & Operations Research, **40**(8):1953–1962 (2013). (Cited on page 15.)
- [51] Gilbert, K.C. and Hofstra, R.B. *A new multiperiod multiple traveling salesman problem with heuristic and application to a scheduling problem*. Decision Sciences, **23**(1):250–259 (1992). (Cited on page 15.)
- [52] Delling, D. *et al.* *Engineering route planning algorithms*. In *Algorithmics of large and complex networks*, pages 117–139. Springer (2009). (Cited on page 15.)
- [53] Prins, C. *A simple and effective evolutionary algorithm for the vehicle routing problem*. Computers & Operations Research, **31**(12):1985–2002 (2004). (Cited on page 16.)
- [54] Min, H. *The multiple vehicle routing problem with simultaneous delivery and pick-up points*. Transportation Research Part A: General, **23**(5):377–386 (1989). (Cited on page 16.)
- [55] Ombuki, B. *et al.* *Multi-objective genetic algorithms for vehicle routing problem with time windows*. Applied Intelligence, **24**(1):17–30 (2006). (Cited on page 16.)
- [56] Falcon, R. *Towards Fault Reactiveness in Wireless Sensor Networks with Mobile Carrier Robots*. Ph.D. thesis, PhD Dissertation, University of Ottawa (2012). (Cited on page 16.)
- [57] Lau, H.C. *et al.* *Application of genetic algorithms to solve the multidepot vehicle routing problem*. Automation Science and Engineering, IEEE Transactions on, **7**(2):383–392 (2010). (Cited on page 17.)

- [58] Hsu, C.I. *et al.* *Vehicle routing problem with time-windows for perishable food delivery*. Journal of food engineering, **80**(2):465–475 (2007). (Cited on page 17.)
- [59] Coello, C.C. *et al.* *Evolutionary algorithms for solving multi-objective problems*. Springer Science & Business Media (2007). (Cited on page 17.)
- [60] Hwang, C.L. and Masud, A.S.M. *Multiple objective decision making—methods and applications: a state-of-the-art survey*, volume 164. Springer Science & Business Media (2012). (Cited on page 18.)
- [61] Kennedy, J. *Particle swarm optimization*. In *Encyclopedia of machine learning*, pages 760–766. Springer (2011). (Cited on page 19.)
- [62] Roy, R. *et al.* *A Novel Particle Swarm Optimization Algorithm for Multi-Objective Combinatorial Optimization Problem*. International Journal of Applied Metaheuristic Computing, **2**(4):41–57 (2011). (Cited on page 19.)
- [63] Chen, W.N. *et al.* *A novel set-based particle swarm optimization method for discrete optimization problems*. Evolutionary Computation, IEEE Transactions on, **14**(2):278–300 (2010). (Cited on page 19.)
- [64] Abdoun, O. and Abouchabaka, J. *A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem*. arXiv preprint arXiv:1203.3097 (2012). (Cited on pages 20 and 22.)
- [65] Blicke, T. and Thiele, L. *A Mathematical Analysis of Tournament Selection*. In *ICGA*, pages 9–16 (1995). (Cited on page 20.)
- [66] Soremekun, G. *et al.* *Composite laminate design optimization by genetic algorithm with generalized elitist selection*. Computers & Structures, **79**(2):131–143 (2001). (Cited on page 20.)
- [67] Banzhaf, W. *et al.* *Genetic programming: an introduction*, volume 270. Morgan Kaufmann San Francisco (1998). (Cited on page 22.)

- [68] Kirkpatrick, S. *Optimization by simulated annealing: Quantitative studies*. Journal of statistical physics, **34**(5-6):975–986 (1984). (Cited on page 24.)
- [69] Helsgaun, K. *General k -opt submoves for the Lin–Kernighan TSP heuristic*. Mathematical Programming Computation, **1**(2-3):119–163 (2009). (Cited on pages 24 and 97.)
- [70] Gendreau, M. *et al.* *A tabu search heuristic for the vehicle routing problem*. Management science, **40**(10):1276–1290 (1994). (Cited on page 24.)
- [71] Labassi, R. *et al.* *Quantitative Risk Management: a Survey of Adaptive Approaches to Risk Management for Information and Communication Systems*. International Journal of u-and e-Service, Science and Technology, **8**(5):105–128 (2015). (Cited on pages 25 and 26.)
- [72] Tan, X. *et al.* *Using hidden markov models to evaluate the real-time risks of network*. In *Knowledge Acquisition and Modeling Workshop, 2008. KAM Workshop 2008. IEEE International Symposium on*, pages 490–493. IEEE (2008). (Cited on page 25.)
- [73] Keramati, M. and Akbari, A. *An attack graph based metric for security evaluation of computer networks*. In *Telecommunications (IST), 2012 Sixth International Symposium on*, pages 1094–1098. IEEE (2012). (Cited on page 25.)
- [74] Elkhodary, A. and Whittle, J. *A survey of approaches to adaptive application security*. In *Proceedings of the 2007 International Workshop on Software Engineering for Adaptive and Self-Managing Systems*, page 16. IEEE Computer Society (2007). (Cited on page 26.)
- [75] Ryan, J.J. and Ryan, D.J. *Performance metrics for information security risk management*. IEEE Security & Privacy, (5):38–44 (2008). (Cited on page 26.)
- [76] Baker, W.H. *et al.* *Necessary measures: metric-driven information security risk assessment and decision making*. Communications of the ACM, **50**(10):101–106 (2007). (Cited on page 26.)

- [77] Coello, C.A.C. *et al.* *Evolutionary algorithms for solving multi-objective problems*, volume 242. Springer (2002). (Cited on pages 28 and 96.)
- [78] Sierra, M.R. and Coello, C.A.C. *Improving PSO-based multi-objective optimization using crowding, mutation and e-dominance*. In *Evolutionary multi-criterion optimization*, pages 505–519. Springer (2005). (Cited on page 28.)
- [79] Krink, T. and Løvbjerg, M. *The lifecycle model: combining particle swarm optimisation, genetic algorithms and hillclimbers*. In *Parallel Problem Solving from Nature—PPSN VII*, pages 621–630. Springer (2002). (Cited on pages 78 and 96.)
- [80] Golombic, M.C. *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier (2004). (Cited on page 95.)
- [81] Jaskiewicz, A. *Genetic local search for multi-objective combinatorial optimization*. European journal of operational research, **137**(1):50–71 (2002). (Cited on page 97.)
- [82] Lust, T. and Teghem, J. *Two-phase Pareto local search for the biobjective traveling salesman problem*. Journal of Heuristics, **16**(3):475–510 (2010). (Cited on page 97.)
- [83] Paquete, L. and Stützle, T. *On the performance of local search for the biobjective traveling salesman problem*. In *Advances in Multi-Objective Nature Inspired Computing*, pages 143–165. Springer (2010). (Cited on page 97.)
- [84] Durillo, J.J. and Nebro, A.J. *jMetal: A Java framework for multi-objective optimization*. Advances in Engineering Software, **42**(10):760–771 (2011). (Cited on page 97.)
- [85] Lust, T. and Jaskiewicz, A. *Speed-up techniques for solving large-scale biobjective TSP*. Computers & Operations Research, **37**(3):521–533 (2010). (Cited on page 97.)