Configuring Multi-Objective Evolutionary Algorithms for Design-Space Exploration of Wireless Sensor Networks[*]

Majid Nabi[1], Milos Blagojevic[2], Twan Basten[1,2], Marc Geilen[1], Teun Hendriks[2]
[1]Department of Electrical Engineering, Eindhoven University of Technology, the Netherlands
{m.nabi,a.a.basten,m.c.w.geilen}@tue.nl
[2]Embedded Systems Institute, Eindhoven, the Netherlands
{milos.blagojevic,teun.hendriks}@esi.nl

ABSTRACT

Wireless sensor networks (WSNs) consist of numerous sensor nodes with several possible configurations for each node. As there are a lot of nodes in a typical WSN, each with its own set of configurations, the number of configurations for the network as a whole is huge and the design space is extremely large. The configuration of a WSN has a strong effect on the quality of services of running applications and the performance of the WSN. Multi-objective evolutionary algorithms (EAs) are well suited to explore the trade-offs in a WSN design space. However, an EA has many configuration parameters in itself. This paper presents several guidelines for configuring a multi-objective EA for design space exploration, given a specification of the WSN to be configured and a time budget available for analysis. We demonstrate the effectiveness of these guidelines on a specific type of WSN that uses a gossip strategy for disseminating data over the network.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design; C.4 [**Performance of Systems**]: Performance attributes; Reliability, availability, and serviceability

General Terms

Algorithms, Performance, Theory.

Keywords

Wireless sensor networks, design-space exploration, genetic algorithms, quality of service, trade-off analysis

1. INTRODUCTION

Wireless sensor networks are used for various applications such as in health care, fire detection, disaster management, and agriculture. Sensor nodes are designed for each application with specific facilities but usually these sensor nodes are tiny and low-cost. Wireless technology makes these networks relatively easy to deploy. However, they also suffer limited lifetime due to the limited battery capacity. There is a lot of research on the physical design of nodes and various network protocols like low-power efficient MAC layers. Typically, a protocol is designed for a specific category of applications to meet its specific constraints. Every WSN at application level considers some specific quality of service (QoS) metrics such as latency, with constraints and optimization goals for each metric.

In a typical WSN, every sensor node has several controllable parameters (like transmission power) and so it can be configured in several ways. There might also be different types of sensor nodes with specific parameters. In addition, the WSN may have network-level configurable parameters (like TDMA frame size). As there are lots of sensor nodes in a typical WSN and every node can be configured with several settings, a WSN has an extremely large number of possible configurations. Each configuration for the network consists of specific configurations for each node as well as a configuration for network-level parameters.

Every configuration affects the quality of considered metrics and selecting the best configuration for the network is of high importance. In a typical application, there might be several QoS metrics and in many cases the various metrics are in conflict with each other. This means that improving some metrics through configuring nodes in some way might worsen others. So there are trade-offs between QoS metrics and because of these trades-offs there can be several configurations that all are Pareto points [16] in the multi-dimensional objective space. As at each time, we cannot consider more than one configuration for the WSN, we may consider some cost function to select a setting among the Pareto points. But the main problem is to find the set of Pareto points.

As there is a huge number of possible configurations for a WSN, an exhaustive search in the configuration space is not feasible. So, using a heuristic multi-objective search method to find near-optimal configurations is reasonable. Evolutionary algorithms (EA), typically in the form of genetic algorithms (GA), are well-known search methods that can be used for WSN design-space exploration (DSE). The main challenge in using a GA is that there is no warranty to reach optimal results in a predictable time. However, a GA has many configuration parameters itself. In this paper, we present guidelines for configuring a GA for a given WSN DSE problem and time budget, aiming to achieve good quality Pareto points within this time budget.

Chen et al. [5] give an overview of recent approaches in QoS for sensor networks. In several papers, the trade-offs between some specific QoS objectives are investigated and some scenarios are proposed to reach best values for a metric considering some constraints on others. Many of these focus on the architecture to deal with the trade-offs. In [4], the trade-off between latency and lifetime as two objectives is analyzed. The result is the optimum number of sensing nodes to reach the best latency while considering efficient energy consumption. In [7], the cost versus the quality is considered to find the proper level of cooperation between sensor nodes. The trade-off between security of a WSN and lifetime performance is analyzed in [15].

Hoes et al. [9] present a qualitative model as well as some quantitative formulae for two specific applications to calculate values for metrics for a given WSN configuration. They also propose an effective compositional method for finding the best solutions to configure the network. The method is limited to a specific type of networks with a tree structure for data routing to a central station or sink, and the functions for calculating metric values should be monotone.

None of the mentioned techniques can be generalized to arbitrary WSN DSE problems. Multi-objective evolutionary algorithms however provide a versatile tool that can be tuned to a wide variety of DSE problems. So in many cases, they are a proper choice. In [11], a GA is used to set the operational mode of every sensor node in a WSN. The paper defines a weighted linear fitness function for a specific WSN design and tries to maximize this linear function by use of a GA. In [3], a bee colony inspired evolutionary algorithm is used for run-time optimization. The elite selection is done in the sink node and genetic operations, including crossover and mutation are done in each sensor node.

We propose general guidelines for using GAs in design space exploration and configuration of WSNs. We apply those guidelines to a gossip-based WSN that has a lot of non-deterministic behavior owing to its gossiping strategy, which cannot be handled easily in a guided search strategy.

The next section formulates the design space for typical WSNs. Guidelines for applying a GA to the DSE of a WSN are proposed in Section 3. In Section 4, a specific gossip-based WSN, its parameters, and its QoS metrics are described, to serve as a case study. Section 5 shows the experimental results and finally Section 6 concludes the paper.

2. THE WSN DESIGN SPACE

In this section, we formalize the design space for a WSN in a generic way. The design space consists of two related parts, the configuration space and the space of QoS metrics of interest.

Every WSN has parameters that define the configuration space. Each parameter can hold values in a specific range or domain which is determined by the users based on application requirements or by the hardware characteristics of sensor nodes. If such a domain is continuous, we assume that some discrete approximation has been defined. In practice, this is not really a limitation, because a small set of discrete values per parameter already defines a large set of operating points of the WSN as a whole. The discrete set of possible values for parameter p is called quantity q_p. In a typical WSN, there are two general categories of parameters, (global) network-level parameters (P_G) and (local) node-level parameters (P_L).

Global parameters refer to a set of parameters which have to be set for the network as a whole. As an example, we can mention the frame length (the TDMA period) of a TDMA-based MAC protocol. All nodes use the same frame length and so this parameter is global for all nodes in the network. We define the *network-level configuration space* (δ_G) as a given subset of the Cartesian product of all quantities for the network-level parameters. So $\delta_G \subseteq \prod_{i=1}^{|P_G|} q_i$ is the network-level configuration space and $\overline{c}_G \in \delta_G$ is a possible configuration for network-level parameters. We consider only a subset of all combinations because it is possible that some parameter combinations are not valid.

On the other hand, the node-level parameters can be configured for each node individually. Transmission power of the wireless transmitter is an example of this kind of parameter. In general, there could be various sensor-node types in a WSN, where each node type has its specific parameters and values for them. Nodes can be different based on their different functionalities in the network and quantities for their parameters, or due to various hardware specifications or different sensor types. For example, some nodes can only have a routing or data storage role without any sensor facilities. Also, there might be some more powerful nodes in terms of processing speed or battery capacity for data aggregation and infrastructure support.

Suppose that we have H different node types in a WSN. Let P_L^k be the set of controllable parameters for node type k in which $1 \leq k \leq H$. A *node-level configuration space* for node type k is a subset of the Cartesian product of all quantities existing in that node type. So we let $\delta_L^k \subseteq \prod_{i=1}^{|P_L^k|} q_i$ be the set of feasible configurations for every node in the network, with q_i the quantity corresponding to parameter i in set P_L^k. $\overline{c}_L^k \in \delta_L^k$ is called a *node-level configuration*.

The configuration space for a WSN with N sensor nodes can be created by the Cartesian product of the node-level configuration spaces of all nodes and the network-level configuration space.

$$\delta = (\prod_{i=1}^{H}(\prod_{k=1}^{K_i} \delta_L^i)) \times \delta_G \quad , \quad \sum_{i=1}^{H} K_i = N, \qquad (1)$$

where K_i denotes the number of nodes of type i. For $H > 1$, we have a *heterogeneous* network which consists of several node types. If the number of node types H is one, we have a *homogeneous* network. In such a network, the configuration space would be $\delta = (\delta_L)^N \times \delta_G$. As an example, Fig. 1(a) illustrates the quantities for two network-level and two node-level parameters in our WSN case study. We suppose a homogeneous network with only one node type for simplicity. The number of all configurations for this network with these four parameters and N nodes would be $8^N \times 9$. In reality, the number of parameters and their possible values can be more than in this example. From the example, however, it can already be seen that for a medium size WSN with 100 nodes, the number of configurations is extremely large and finding the best configurations through an exhaustive method is typically not feasible. So, some heuristic methods may be needed for finding (near-)optimal solutions among these possible configurations.

For any WSN DSE problem, there are some application level QoS metrics that are of interest to the designer or user. The goal of the design-space exploration is to optimize all metrics as far as possible while potentially taking some constraints into account. This leads in general to a trade-off
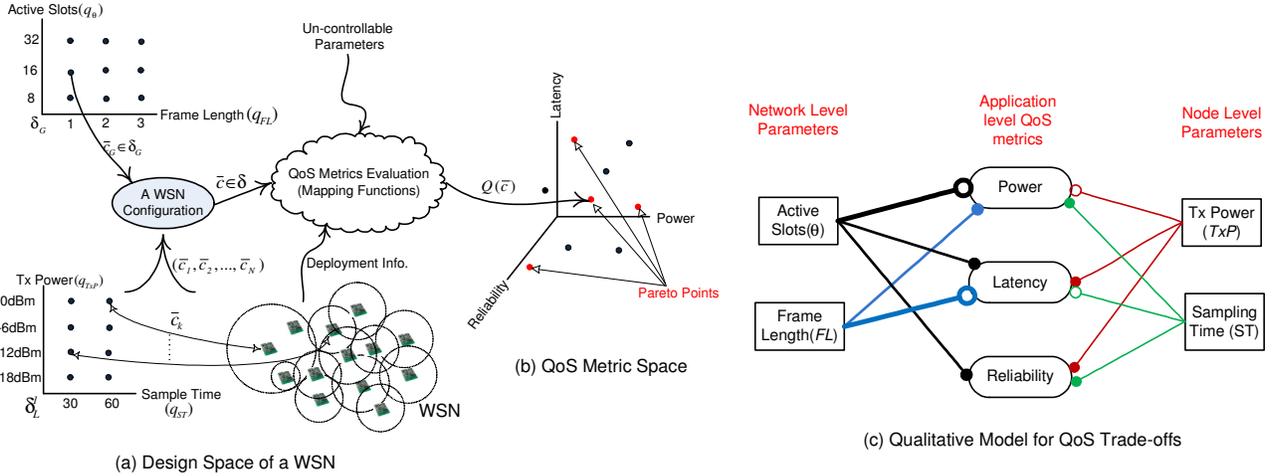
Figure 1: Design space of a WSN and its QoS trade-offs Model.

between different metrics. We can define metrics at any level in a network, such as node-level metrics, MAC-level and network-level metrics and also application-level metrics. We are interested to optimize application-level metrics of the deployed WSN. Ultimately, only one configuration for the WSN needs to be selected. However, as long as the relative importance of the various metrics is not known, we are interested in finding the trade-offs between the metrics. Therefore, in this paper, we focus on trade-off analysis. Fig. 1(c) illustrates the trade-offs between QoS metrics in terms of various parameters. A filled circle represents a positive relationship and an open circle a negative relation. The line width illustrates the strength of the effect of the corresponding parameter on the target metric.

Let us define M as the set of QoS metrics for the WSN with d elements ($|M| = d$). Each metric can get a value according to the applied configuration and so for each metric $m \in M$, there is a mapping function $f_m : \delta \rightarrow Q_m$ which maps the configuration to a value in the domain (Q_m) of metric m. The mapping function can be a concrete formula or it could be the result of a simulation for the desired configuration at any abstraction level. For a GA, we prefer mapping functions that can be evaluated fast. A quality vector $\overline{Q}(\bar{c}) = (f_1(\bar{c}), f_2(\bar{c}), \ldots, f_d(\bar{c}))$ is a vector of discrete values of all QoS metrics for the configuration $\bar{c} \in \delta$.

The main goal of trade-off analysis is indeed finding Pareto points among all possible quality vectors. A Pareto-optimal point is a configuration which is not dominated by any other point in the design space. Point p_1 dominates point p_2, denoted $p_1 \succeq_d p_2$, if p_1 is not worse than p_2 in terms of any quality metric. Moreover, a Pareto-optimal set is the set of all Pareto points in the decision space (Pareto points shown in Fig. 1(b) as an example). Generating the complete Pareto-optimal set is practically infeasible because of the huge design space. So, an approximated Pareto set is expected to be generated by the DSE process. The quality of the approximated Pareto set is determined by the closeness to the exact Pareto set.

To quantify the quality of an approximated Pareto set, we use the $binary\ \epsilon - Indicator$ criterion [18] which uses the $\epsilon - dominance$ relation. A point p_1 is said to $\epsilon - dominate$ another point p_2, denoted $p_1 \succeq_\epsilon p_2$, if we can multiply each objective value of p_2 by a factor ϵ and it is still dominated by

p_1 (where less is supposed to be better for objective values). The $binary\ \epsilon - Indicator$ as defined below is actually the minimum factor ϵ such that any point in Pareto set B is $\epsilon - dominated$ by at least one point in Pareto set A.

$$I_\epsilon(A, B) = \inf\{\epsilon \in \mathbb{R} \mid \forall\ b \in B\ \exists a \in A\ :\ a \succeq_\epsilon b\} \quad (2)$$

To assess the quality of an approximated set, $I_\epsilon(A, Ref)$ is calculated in which Ref is the reference Pareto set.

3. A GA-BASED APPROACH TO WSN DSE

In this section, we first introduce GAs and then we describe our guidelines for implementing and configuring a GA for WSN design-space exploration.

3.1 Genetic Algorithms

Genetic algorithms are the most important class of EAs that use some biologically inspired techniques to approximate optimal solutions for optimization problems.

In a genetic process, an $individual$ is a candidate problem solution and its representation depends on the problem at hand. A genetic process begins with an $initial\ population$ of individuals. Then, in steps referred to as $generations$, some individuals are selected based on a specific $fitness\ function$ to form a new generation. This new generation is created via $genetic\ operations$. The most common genetic operations are $recombination$ and $mutation$. In recombination, parents participate in creating one or more children and the children inherit each part of their specifications from one of their parents. To avoid convergence to local optima, mutation is used to change individuals with a certain probability.

More formally, a genetic process can be presented as an iterative function $\Psi : 2^\delta \rightarrow 2^\delta$ with $\Psi(\rho_i) = \rho_{i+1}$, where i is a generation index and $\rho_k \subseteq \delta$ is the population in generation k. The process starts with an initial population (ρ_0) and continues until a maximum number of generations G has been generated. Finally, the results of the process are the Pareto points of the population in the final generation (ρ_G). We can decompose the genetic process into two main parts; the $variator$ and the $selector$.

$$\Psi = \Psi_{var}\ o\ \Psi_{sel}\ :\ \Psi_{var}(\rho_i) = \rho_i'\ ,\ \Psi_{sel}(\rho_i' \cup \rho_i) = \rho_{i+1}\ (3)$$

The $variator$ performs the genetic operations to create new individuals (ρ_i') as well as the QoS metrics evaluation for

the individuals. Also the initial population generation is included in this component. Actually, this part depends on the specification of the problem that is solved by the GA. Selection is done in the *selector* part in which fitness values are used to select some individuals to create the next generation (ρ_{i+1}). This part is mainly independent of the problem at hand and considers the QoS metric values attached to every individual.

In our experimental evaluation, we use an adapted version of SPEA2 [19] for the *selector* part. SPEA2 assumes a fixed population size and uses a fine-grained fitness assignment strategy to select best configurations to keep as the population. The *strength* of every individual is calculated as the number of individuals it dominates. Subsequently, the *fitness* value for every individual is determined by summing the strength of its dominators. If the fitness value for an individual is zero, that solution is a Pareto point and will be selected (assuming the population is sufficiently large). An estimate of the density of individuals in the search space is used to add some dominated individuals when the number of Pareto points is less than the population size. If the number of Pareto points exceeds the population size, a truncation method is proposed to remove some Pareto points from the population, based on the minimum distance to other individuals. We chose SPEA2 because it has been shown [19] to have in general better performance than other multi-objective evolutionary algorithms.

3.2 Configuring a GA for WSN DSE

For DSE in the WSN area, on the one hand, we have a given WSN with a certain specification and on the other hand, we have a GA with a specific selection method that has itself some controllable parameters. Setting the GA parameters properly, according to the specification of a given WSN DSE problem, can lead us faster to better results. At first, we list the WSN specifications that affect the decision about the GA configuration and then we describe the parameters in a GA that we can control.

According to the terminology we used in the design-space formulation, d, N, P_G, δ_G, H, P_L and δ_L all contribute to the WSN DSE specification. Also T_e, which denotes the time needed to evaluate metrics for a specific configuration (individual) plays an important role in our decisions about the genetic process. It mainly depends on the abstraction level of the underlying WSN models used, as well as on other specifications of the network like the network size.

On the other side, in a genetic process, in general, there are some parameters which are adjustable for the specific problem at hand. It is interesting to consider the specification of the WSN in decision making about these parameters. G as the maximum number of generations, $|\rho_i| = \alpha$ as the population size, $|\rho'_i| = \lambda$ as the number of new individuals in each generation and the mutation probability (P_m) are some configurable parameters of a GA. Also the method of initial population selection and the recombination and mutation mechanisms (Ψ_{var}) can be decided based on the WSN DSE specification.

As mentioned before, we are going to configure the genetic process for the given WSN DSE problem to achieve Pareto points with higher quality in less time. Indeed, we have a meta trade-off between analysis time and quality of Pareto points in the final result (ρ_G). Fig. 2(b) depicts this trade-off. A normal arrow head expresses that it is not always
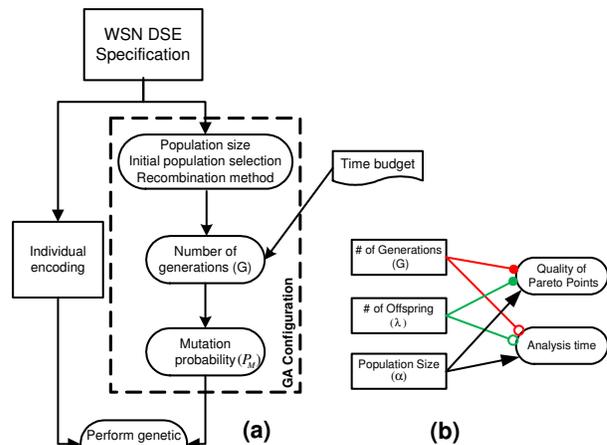
Figure 2: (a) Strategy for configuring a GA for WSN DSE (b) Trade-offs between time and quality of Pareto points in GA.

clear whether an effect is positive or negative. With more generations, we will generally achieve better points but it takes more time. On the other hand, proposing more configurations in each generation increases the chance of finding better points in each generation. But again this needs more time especially when the evaluation of a configuration takes relatively much time because we have to evaluate QoS metrics and fitness for every new configuration. Also the population size α affects the quality of results and the analysis time consumption. For this parameter, it is not always clear how it affects quality and analysis time.

Fig. 2(a) shows our strategy to prepare a GA for a specific WSN. First, the WSN DSE problem should be implemented; in particular the WSN configurations must be implemented as the individuals in the genetic process. Also, we need to decide about the concrete instances of the operations in the GA and conditions for the termination of the process. The remainder discusses these steps.

3.3 Problem Implementation

In a GA for solving a WSN DSE problem, every individual represents a configuration for the network, where the corresponding metric values are attached to it. The individual corresponding to configuration $\bar{c} \in \delta$ is $\overline{ind} = \bar{c}.\bar{Q}(\bar{c})$. For our implementation, we use integer coding to represent the configuration part of the individuals. A configuration in our implementation is a vector of $N + 1$ integer values.

$$\bar{c} = [k_1, ..., k_{N+1} | k_i \in \mathbb{N}(|\delta_L^{T(i)}|), k_{N+1} \in \mathbb{N}(|\delta_G|)] \quad (4)$$

where $T(i)$ returns the type of node i in the WSN and $\mathbb{N}(x)$ stands for positive natural numbers less than or equal to x. Each element (k_i) of the first N elements of the vector is an integer value that refers to a configuration from the node-level configuration space of the node type for node i; k_{N+1} refers to a network-level configuration. So, we generate the configuration space for each node type and use an index in every individual to this space for every node. Because there are only a few node types in a typical WSN and the configuration space for every node type is relatively small, it is efficient to keep all configurations in memory. Fig. 3 shows a simple example of an encoded individual for the homogeneous WSN of our running example.
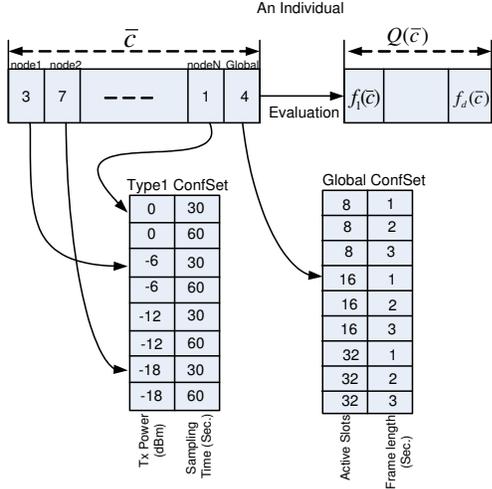
Figure 3: Individual coding for a WSN.

3.4 Initial Population and Population Size

Traditionally, some configurations are selected purely randomly as the initial population. However, for WSN DSE problems, network-level parameters typically have a stronger effect on QoS metrics than node-level parameters. Settings for a specific node in the network cause small variations around the point set by the global configuration. So, we try to cover all possible network-level configurations in the initial population. We therefore propose to choose the initial population as follows.

Guideline 1 (Initial population selection) *Let ρ_0 be the initial population of the GA such that for every network-level configuration, there is at least one individual with that configuration as its network-level configuration. In other words, $\forall x \in \mathbb{N}(|\delta_G|) \; \exists \overline{ind} \in \rho_0 \; : \; \overline{ind}.\overline{c}(N+1) = x$. Set the number of initial individuals $\alpha_0 \simeq c_0|\delta_G|$ where $c_0 \geq 1$ specifies the occurrence count of one network-level configuration.*

A higher c_0 yields a higher chance for network-level configurations to remain in the population but with more time needed for QoS evaluation. The number of QoS metrics (d) can be used as a proper value for scaling c_0.

Experiments on our WSN DSE case study show that the best population size is strongly related to the number of Pareto points which itself depends among others on the dimensionality of the metric space. If the number of Pareto points in a generation is much more than the population size, we have to truncate many Pareto points, which causes some loss in the final quality. On the other hand, if the population size is much larger than the number of Pareto points, we would have many dominated points in the population and since they may participate in the genetic operations, the final results would become worse. So, we suggest to have a variable population size which is around the number of Pareto points in every generation. But if the population size becomes very large, the time consumed by the *selector* part will also be very large. So, we attach a maximum value for the population size.

Guideline 2 (Population size) *We suggest to set the population size of every generation of the GA determined on the fly such that $\alpha_i = min\{(1+c_1)Par(G_i) \, , \, \alpha_{max}\}$ where c_1 is some constant, $Par(G_i)$ is the number of Pareto points in generation i and α_{max} is the maximum population size.*

This means that the population size for every generation is set to the number of Pareto points in that generation plus a percentage to keep some good dominated points. This will be continued until the population size exceeds α_{max}. To specify the α_{max} based on the given WSN DSE problem, we should consider the expected number of Pareto points. The number of Pareto points mainly depends on the dimensionality of the metric space and possible correlation between the metric values.

Guideline 3 (Maximum population size) *Adapt the maximum population size of the GA to the WSN DSE specifications such that $\alpha_{max} = c_2|\delta_G|$. Scale the constant c_2 with the number of expected Pareto points in a set of WSN configurations.*

Due to the irregularity of WSN DSE problems, the expected number of Pareto points may be difficult to capture in a formula. However, a good indication can typically be obtained via a small number of experiments.

Regarding the number of new individuals in every generation (λ), the network-level configuration size can be an appropriate minimum value for λ to make sure that we obtain good coverage of all network-level configurations. However, we have to consider the evaluation time of our quantitative models which is actually the main time consuming part of the genetic algorithm process. With a given time budget for DSE, having a larger number of new individuals in every generation decreases the number of generations that can be explored.

3.5 Genetic Operations

Recombination is done in any generation to propose new individuals, of which hopefully some are better than the current population for the WSN. A commonly used method is a uniform crossover for two individuals in which node-level configurations between two individuals are swapped with some probability. An important aspect in the recombination is selecting the parents among the population to mate. For now, we use the mating parents selection of SPEA2 which is based on a tournament between randomly selected individuals from the population, and which has a demonstrated good quality performance in multi-objective optimization. As future work, we plan to investigate whether we can use information about the WSN DSE problem to improve the parent selection.

Mutation is done to prevent the genetic process from converging to local optima. Every new individual in the output of the recombination phase is mutated. To do that, the configuration assigned for every node in the individual will be changed to an arbitrarily chosen configuration with a low probability (P_m).

The most important aspect to consider is that the probability of this operation affects the convergence speed of the GA. It is concluded in [12] that reducing the mutation probability exponentially over time increases the performance of the GA. Moreover, this conclusion has been confirmed in [14], which also shows dependencies of this probability on some aspects from the problem specification like dimensionality of the configuration space and population size. We involve also the maximum number of generations in the decision about the value of this probability. So we propose the following guideline to generate a variable mutation probability.

Guideline 4 (Mutation probability) *For every newly generated individual at generation t, mutate all its node-level configurations with the following probability.*

$$P_m(t) = \frac{c_3 e^{-c_4 t/G}}{\lambda \sqrt{N}} \quad t = 0, ..., G-1 \qquad (5)$$

Experiments show that the dimensionality d is important for decision about the constants c_3 and c_4. For higher dimensionality the constant c_3 needs to be higher while the constant c_4 is better to be lower. Eqn. 5 shows how to adapt the mutation probability for different network sizes (N), the number of offspring in every generation (λ), and the number of generations which is going to be done (G).

3.6   Process Termination Criterion

In this step, we should decide about the number of generations we want to perform. This can be done by considering the time budget and the time complexity of the computations that are needed in every generation. First, we identify the main time-consuming steps in a GA. T_{var} stands for the time needed by the *variator* step in all generations. It includes the time for evaluating the QoS metrics of the individuals (T_e), which is the main part, as well as the time for genetic operations (T_{op}).

$$T_{var} = T_e(\lambda G + \alpha_0) + T_{op} G \qquad (6)$$

The total time needed by the *selector* part in a generation (T_{sel}) has time complexity $\mathcal{O}((d(\lambda + \alpha))^2)$. This time consumption is considered to decide about the maximum population size. Finally, the total time of the genetic process is T, which is the sum of the *variator* and *selector* components.

$$T = T_{var} + GT_{sel} = T_e(\lambda G + \alpha_0) + G(T_{op} + T_{sel}) \qquad (7)$$

Guideline 5 (GA termination) *Given a time budget for DSE ($T_{\max}$), the number of generations to be explored is as follows:*

$$G = \frac{T_{max} - \alpha_0 T_e}{(\lambda T_e + T_{op} + T_{sel})} \qquad (8)$$

T_e has a dominant effect on the time needed by the process and depends on the abstraction level of the WSN models at hand. However, if the evaluation time for a single configuration is relatively high, we should also be careful about the number of new individuals (λ) and also their quality. Besides deciding the number of generations G based on the available time budget, it is possible to apply some constraints for the quality of the Pareto points obtained in each generation as a condition for terminating the genetic process. We use the time budget and compute the maximum number of generations because this parameter affects our decision about the mutation probability. Investigating the use of additional stop criteria based on the quality of the results obtained up to a certain generation is left for future work.

4.   CASE STUDY: A GOSSIP-BASED WSN

The WSN considered to illustrate our configuration guidelines is based on data gossiping. In this type of WSN, the system is supposed to be a medium for data dissemination.

4.1   The System

A *MyriaNed* [17] wireless node is used as sensor node. It uses a *Nordic* chip as transceiver which broadcasts radio packets of size 32 Bytes with a 2.4GHz carrier frequency and data rate up to 2Mbps. DG-MAC [2] is used as a low power MAC layer which is a TDMA-based protocol well suited for running a data gossiping algorithm on top of it. In this protocol, each MAC frame period is divided into some equal slots. Each node is active in some slots and inactive in others to reduce power consumption. A node will transmit in exactly one dedicated slot which is unique in a 2-hop neighborhood and listens in others. Finally, every node has another transmit slot outside the active part for synchronization purposes.

A gossiping algorithm called *SharedStates* [13] is used on top of the MAC layer. In this algorithm, every node has a limited memory space, the data cache, to store some data items. In every round, some data items are selected in a random fashion to propagate in the transmission slot. The number of these items is limited to the size of the packet. Also a node which is equipped with a sensor adds its own sensed data item to the data cache. We are using an illustrative temperature mapping application with a 10×10 grid topology. Every node is equipped with a temperature sensor and we suppose an adequate data cache size to store the whole temperature map in each node.

4.2   Parameters and QoS Metrics

Figure 1 depicts an overview of the parameters and metrics in our WSN case study. We have two network-level parameters and also two node-level parameters. For every node, we can set a transmission power TxP for the radio and it determines the radio range. The other node-level parameter is the sampling time (ST). This parameter determines the time distance between two subsequent sensing actions for the sensor node; this means that in every ST seconds a node creates a new data item.

We also consider two parameters for the network level. These parameters are the frame length (FL) and the number of active slots per frame (θ) in TDMA scheduling. There are several other parameters like environmental parameters which are out of our control. We are only interested in controllable parameters, because these parameters constitute the configuration space. It may be interesting to consider robustness against changes in uncontrollable parameters as a quality metric for the WSN, but for now we leave this as future work.

For the temperature mapping application, we consider three QoS metrics. These metrics relate to the commonly considered dimensions of time, reliability of data transmission, and lifetime (power consumption) and are therefore typical for many applications of WSNs.

Typically in WSNs the battery capacity of sensor nodes is limited and a node will die when its energy storage depletes. We define the *lifetime* of a WSN as the time to first node failure in the network. If all nodes have equal battery capacity, the most power consuming node will fail first and the time of this failure is the network *lifetime*. Of course, the network may still function after one node dies, but for now, we assume that this results in a new situation that needs to be re-analyzed, possibly by some reconfiguration methods.

Latency is the second metric, which we define as the average age of the maps in all nodes. The age of a map is the average age of data items in the map and the age of each data item is the difference between the current time and the sampling time of the data item. The final quality of service metric is *reliability*. It is defined as the average percentage

of data items that arrives in order. Since the behavior of *SharedStates* as a gossiping algorithm uses some randomization in broadcasting data items, there is a chance that a later items arrives before an earlier one.

4.3 WSN Model for QoS Trade-off Analysis

In our application, there are several trade-offs for the three QoS metrics. Fig. 1(c) shows a qualitative model for the network and depicts all trade-offs. Increasing the frame length causes a higher latency in dissemination of data items but it has a good effect for lifetime. Note that for latency less is better but for lifetime more is better. Thus lifetime and latency are in conflict in terms of frame length.

The number of active slots per frame defines the active or power consuming part of a frame and so increasing it leads to decreasing lifetime. On the other hand, having more active slots in a frame leads to less latency and a better reliability. Therefore, latency and reliability are in trade-off with lifetime in terms of parameter θ.

The situation is the same for transmission power. Increasing this parameter has a good effect on latency and reliability but it has a bad effect on lifetime. Finally, sampling time produces a trade-off between reliability and latency as well as between lifetime and latency.

The qualitative model matches with the result of simulation experiments. Our network simulator uses MiXiM [6] on top of OMNET++ [1] and it contains the complete network stack. However, for performing a GA to find the best configurations for the WSN, we need to evaluate every candidate configuration in terms of QoS metrics. As shown in the previous section, the evaluation time is an important factor for decisions about the number of new candidate configurations in each generation and the total number of generations.

The evaluation process could be done at various abstraction levels. To have a precise evaluation, we may use low level abstraction models like simulating the full network stack. But then a simulation for a given network configuration takes too much time to be suitable for DSE.

Indeed, we need a faster evaluation method at a higher abstraction level. Note that for trade-off analysis we need to know the relative values of metrics for every configuration, not necessarily the exact values. For our current exploration, we use a simple model to calculate the value of the metrics: *lifetime*, *latency* and *reliability*.

To estimate network *lifetime* we calculate the power consumption of a node in the gossiping protocol. Eqn. 9 is used to calculate the energy consumption for sensor nodes.

$$Power = \frac{(\theta - 1)E_{rx} + 2E_{tx} + \theta E_{mcu}}{FL} + \frac{E_s}{ST} \qquad (9)$$

where E_{rx}, E_{tx}, E_{mcu} are receiver, transmitter and processor energy consumption per slot, respectively. E_s is the energy needed for a sensing action, which depends on the sensor type. Lifetime of a node is calculated based on its battery capacity and its power consumption. Finally, the minimum lifetime over all nodes gives the network lifetime.

For *latency*, we use the expected delay for a 2D grid topology for gossiping networks stated in [8] as a basic method for latency evaluation. Based on that, the expected number of rounds required to gather an event detected at distance h satisfies h/q with q the link reliability. We calculate the shortest distance of all nodes to one node and based on that we estimate the latency for that node. The resulting value
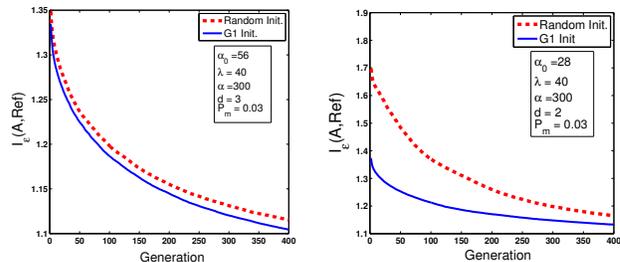
Figure 4: The effect of initial population selection for 3D (left) and 2D (right) optimization.

gives us an upper bound for latency because there might be several shortest paths to reach a specific node. To calculate network latency and the age of a temperature map, the average of latencies is calculated over all nodes.

To estimate the *reliability*, we use a simple model based on the qualitative model (Fig. 1(c)) in which the length of the shortest path between nodes and the sampling time of the source nodes are considered. In each iteration, a node is supposed as a destination and the distances of all other nodes to that node are calculated. When a source node has a bigger sampling time or the distance is longer, the probability of arriving data items in order is higher which means a higher reliability for that link. The average reliability is calculated for every node and the network reliability is the average of calculated reliability for all nodes. Also, the number of active slots improves the reliability.

5. EXPERIMENTS

The described gossiping WSN that we are using as our case study, is a WSN instance, for which there is no known method to explore the design space in a scalable and guided fashion. Thus we used the GA for multi-objective optimization to evaluate the effects of configuring the GA on the quality of the resulting approximated Pareto sets.

We implemented our problem in the PISA framework [10]. In this framework, the *variator* and *selector* parts are completely separated. They communicate with each other through text files. We developed our own *variator* and it is possible to use any *selector* algorithm existing in the PISA library. The SPEA2 algorithm is selected because of its demonstrated performance for multi-objective optimization [19]. We use the $I_\epsilon(A, Ref)$ metric [18] to assess the quality of approximated Pareto set A with respect to reference Pareto set Ref. A lower value shows a better approximation set. Since the exact Pareto set is not practically reachable, we run the GA process for very many generations (5000) and a large λ (200) and assume the resulting set as the reference set Ref. Moreover, as there is much randomness in the GA, we run every method with several *seeds* (32 different values) for the random generator and use the average results over all runs. The processes are run on a PC with a 2.2GHz Core 2 Duo processor, 2GB RAM and running windows XP.

In the WSN design space of our case study, there are 28 network-level configurations ($|\delta_G| = 28$) and 20 different possible setting for every sensor node. In the experiments, the GA process is run for 400 generations with different settings and $I_\epsilon(A, Ref)$ of every generation is shown in curves to reveal the convergence speed of the process. In every generation, the approximated set A is the Pareto subset of
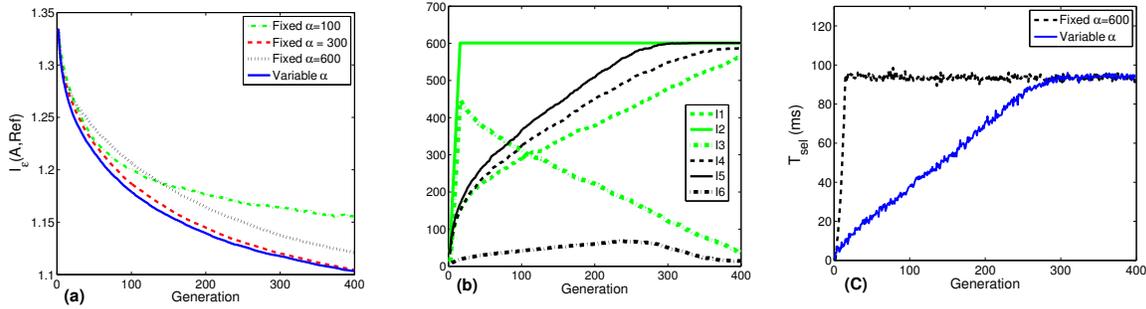
Figure 5: The effect of population size. (a) Quality of approximated Pareto sets for different population size ($\alpha_{max} = 600$ for variable size). (b) I1, I2 and I3 are the number of Pareto points, population size and the number of dominated points in the population for a fixed $\alpha = 600$, respectively. I4, I5, I6 are for a population size determined on the fly with $\alpha_{max} = 600$. (c) Time consumption of the *selector* part.
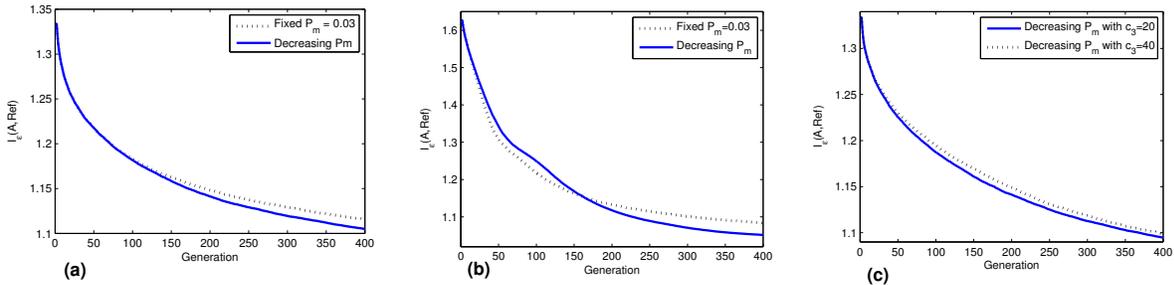
Figure 6: The effect of mutation probability. (a) 3D optimization with a fixed and also a decreasing probability. (b) The quality of the results for 2D optimization. (c) Two GA runs with two different values for the constant c_3 in 3D optimization.

the population in that generation. In some curves, we consider 2D optimization in which the objectives are lifetime and latency whereas reliability is also considered for 3D optimizations.

Figure 4 depicts the effect of initial population selection using *Guideline 1* on the quality of the results. The initial population size is set to 56 ($c_0 = 2$) for 3D optimization and to 28 ($c_0 = 1$) for 2D optimization. In one GA run, a purely randomly selected initial population is used (dotted curves). The solid curves show the quality of the result when we use *Guideline 1* to generate the initial population. The figure reveals that with the same initial population size, the quality of the approximated set is higher when all possible network-level configurations exist in the initial population. Moreover, it can be seen that the improvement is more for 2D optimization. The reason is that in contrast to reliability, the lifetime and latency metrics are strongly dependent on the network-level configuration. These dependencies can be extracted from the qualitative model in Fig. 1(c).

For the rest of the experiments, we always use *Guideline 1* for initial population with the mentioned sizes. Fig. 5 shows the quality of the GA results for various population sizes. We tried three different fixed population sizes (100, 300 and 600) as well as a population size determined on the fly ($\alpha_{max} = 600$) based on *Guideline 2*. For the variable population size, the constant c_1 is set to 0.1. This means that in every generation, the population size is equal to the number of Pareto points plus 10%. The parameter λ is set to 40 for all runs and $\alpha_0 = 56$. All curves are for 3D optimization.

Firstly, the curves with fixed population size shown in Fig.

5(a) illustrate that a big population size does not necessarily improve the quality. Also, a small α generates lower quality results. Comparing the variable α versus the fixed sizes shows that the variable size performs better. This is because for the first generations, there are only a few Pareto points and the population inevitably contains many dominated points. The number of Pareto points in each generation is shown in Fig. 5(b) as well as the population size and the number of dominated points existing in the population. The variable size population consumes less time while it has better results. Fig. 5(c) depicts the time consumed by the *selector* part of the GA for one generation for a fixed population size with $\alpha = 600$ as well as a variable population size with $\alpha_{max} = 600$. Moreover, the total time consumption of the GA for these two settings is 111.48 and 98.36 seconds, respectively. This means that we can run around 50 more generations using *Guideline 2* in the same time budget and thus we receive better results.

For evaluating the role of setting the mutation probability, we run the process with an appropriate fixed probability (0.03) as well as an exponentially decreasing probability according to *Guideline 4* for both 2D and 3D optimization. Fig. 6 shows that the final results of decreasing probability have better quality than the fixed probability version. Note that according to *Guideline 4*, the number of generations G is involved in the mutation probability. So it is expected that the results may be worse for earlier generations. This can be seen in Fig. 6(b) in which the quality of results is shown for 2D optimization. However, the figure shows that the process converges faster to better quality with an exponentially decreasing probability. Actually, this is precisely
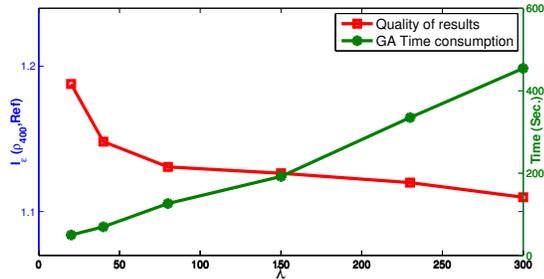
Figure 7: The effect of λ on time and quality.

what *Guideline 4* is aiming at: the best values for mutation probability to reach better results in final generations.

Also, we tried several values for constants c_3 and c_4. Illustrative results for two runs with different c_3 are shown in Fig. 6(c) for 3D optimization. Our experiments suggest that constant c_3 should be decreased when the dimensionality of the DSE problem increases whereas the constant c_4 should be increased.

Finally, *Guideline 5* suggests how to determine the number of generations given a time budget for the genetic analysis. Parameter λ may have a large effect on the possible number of generations that can be analysed within a given budget. To evaluate the effect of λ, we run the GA with several different values for this parameter. Fig. 7 shows the cumulative time consumed by the GA as well as the quality of final results obtained from performing 400 generations. It can be seen that the parameter λ has a strong effect on the time consumption of the GA. However, increasing λ does not have a strong effect on the quality. This confirms our earlier suggestion that λ should not be set too large.

6. CONCLUSIONS AND FUTURE WORK

This paper investigates preparing a genetic algorithm (GA) for exploring the design space of a WSN. It defines the design space for configuring WSNs in a generic way, showing that this space is extremely large in general. GAs are a candidate to explore the multi-objective design space when a scalable and guided search scheme is not available. As the configuration of the GA itself affects the quality of the results that can be obtained in a given time, we propose some guidelines to configure the GA based on the WSN design-space exploration specifications. Initial population selection, population size, mutation probability and process termination criteria are the GA parameters that we investigated. To evaluate our guidelines, we introduced a gossiping WSN, its controllable parameters, our running application, the QoS metrics, and a qualitative model as well as a simple high-level quantitative model. Experiments show that the guidelines lead to better quality configurations for the WSN in less time.

As future work, we plan to investigate the effects of the abstraction level of the quantitative models of WSN QoS. Lower level models based on simulation consume more time but are more accurate. Simple high-level models are less accurate but can be evaluated faster for any given WSN configuration. This may in turn allow us to explore extra generations or more offspring per generation. We also plan to test a hybrid version using models with various abstraction levels in a single run of the GA, high-level abstract models for driving the general direction of the exploration, and low-level accurate models for tuning the results. Another interesting topic for further investigation is the use of quality-driven stop criteria for the genetic process. Finally, we plan to investigate a method for selecting the best parents among the population, that somehow cover the whole range of WSN QoS metric values.

7. REFERENCES

[1] OMNeT++ website. http://www.omnetpp.org.
[2] P. A. Anemaet. Distributed G-MAC: A flexible MAC protocol for servicing gossip algorithms. Master's thesis, Technical University of Delft, the Netherlands, 2008.
[3] P. Boonma and J. Suzuki. MONSOON: A coevolutionary multiobjective adaptation framework for dynamic wireless sensor network. In *Proc. 41st Hawaii Int'l Conf. on System Science*, pages 497–507. IEEE CS, 2008.
[4] F. Bouabdallah, N. Bouabdallah, and R. Boutaba. Analysis of the latency-lifetime tradeoff in wireless sensor networks. In *Proc. IEEE AICCSA*, pages 1076–1081. IEEE, 2008.
[5] D. Chen and P. K. Varshney. Qos support in wireless sensor networks: A survey. In *Proc. Int'l Conf. on Wireless Networks (ICWN 2004)*. CSREA Press, 2004.
[6] A. K. et al. Simulating wireless and mobile networks in OMNeT++ - the MiXiM vision. In *Proc. 1st Int'l Conf. on Simulation tools and techniques for communications, networks and systems and workshops*. ICST, Brussels, 2008.
[7] E. B. et al. Analysis of cost-quality tradeoff in cooperative ad hoc sensor networks. In *Proc. IEEE Int'l Conf. on Communications (ICC)*, pages 112–117. IEEE, 2008.
[8] P. K. et al. Data gathering in a sensor network. Tech. report, Chess Company, the Netherlands.
[9] R. H. et al. Analysing QoS trade-offs in wireless sensor networks. In *Proc. 10-th ACM Int'l Conf. MSWiM 2007*. ACM, 2007.
[10] S. B. et al. PISA- a platform and programming language independent interface for search algorithms. In *Proc. EMO 2003*, pages 494–508. Springer Berlin, 2003.
[11] K. P. Ferentinos and T. A. Tsiligiridis. Adaptive design optimization of wireless sensor networks using genetic algorithms. In *Journal of Computer networks*, volume 51, pages 1031–1051. Elsevier, March 2007.
[12] T. Fogarty and J. Schaffer. Varying the probability of mutation in the genetic algorithm. In *Proc. of 3rd Int'l Conf. on Genetic Algorithms and Applications*, pages 104–109. Morgan Kaufmann Publishers, 1989.
[13] D. Gavidia and M. van Steen. A probabilistic replication and storage scheme for large wireless networks of small devices. In *Proc. 5th IEEE Int'l Conf. Mobile and Ad Hoc Sensor Systems (MASS)*. IEEE, 2008.
[14] J. Hesser and R. Manner. Towards an optimal mutation probability for genetic algorithms. In *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, volume 496, pages 23–32. Springer, 1990.
[15] Z. Karakehayov. Security - lifetime tradeoffs for wireless sensor networks. In *Proc. IEEE ETFA*, pages 646–650. IEEE, September 2007.
[16] V. Pareto. Piccola biblioteca scientifica. *Manual of Political Economy*, pages 795–825, 1906. Translated into English by Ann S. Schwier (1971).
[17] F. van der Wateren. The art of developing WSN applications with myrianed. Tech. report, Chess Company, the Netherlands, 2008.
[18] E. Zitzler, J. Knowles, and L. Thiele. Quality assessment of pareto set approximations. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 373–404. Springer, 2008.
[19] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Proc. EUROGEN Conf. on Evolutionary Methods*, pages 95–100. CIMNE, 2002.