An Evolutionary Framework for Routing Protocol Analysis in Wireless Sensor Networks

Doina Bucur[2,1], Giovanni Iacca[1], Giovanni Squillero[3], and Alberto Tonda[4] *

[1] INCAS[3]
Dr. Nassaulaan 9, 9401 HJ, Assen, The Netherlands
giovanniiacca@incas3.eu
[2] Johann Bernoulli Institute, University of Groningen
Nijenborgh 9, 9747 AG Groningen, The Netherlands
d.bucur@rug.nl
[3] Politecnico di Torino
Corso Duca degli Abruzzi 24, I-10129, Torino, Italy
giovanni.squillero@polito.it
[4] INRA UMR 782 GMPA
1 Avenue Lucien Brétignières, 78850, Thiverval-Grignon, France
alberto.tonda@grignon.inra.fr

Abstract. *Wireless Sensor Networks (WSNs) are widely adopted for applications ranging from surveillance to environmental monitoring. While powerful and relatively inexpensive, they are subject to behavioural faults which make them unreliable. Due to the complex interactions between network nodes, it is difficult to uncover faults in a WSN by resorting to formal techniques for verification and analysis, or to testing. This paper proposes an evolutionary framework to detect anomalous behaviour related to energy consumption in WSN routing protocols. Given a collection protocol, the framework creates candidate topologies and evaluates them through simulation on the basis of metrics measuring the radio activity on nodes. Experimental results using the standard Collection Tree Protocol show that the proposed approach is able to unveil topologies plagued by excessive energy depletion over one or more nodes, and thus could be used as an offline debugging tool to understand and correct the issues before network deployment and during the development of new protocols.*

Keywords: Wireless Sensor Networks, Anomaly Detection, Network Efficiency, Routing Protocols, Evolutionary Algorithms

1 Introduction

The escalation of complexity in modern systems makes proving their correct behaviour an increasingly hard task. *Formal verification techniques* require a human and computational effort that severely limits their applicability. Thus, formal verification methods are used in real-world problems for a subset of the

* All authors contributed equally and their names are presented in alphabetical order.

system's software or hardware components [3], when the task can be artificially constrained or when simplified models are employed, thus significantly impairing the breadth and usability of the results. *Wireless Sensor Networks* (WSNs) are an emblematic example: while each node in a network is a constrained embedded system, the quality of networking depends upon physical topology and environmental conditions. Thus, the number of configurations to test easily skyrockets. Recent related work in the area of formal verification for WSNs [11, 15, 2, 12] succeeds in checking only *qualitative* properties (e.g., assertions), and may only report a small number of faulty topologies.

This paper presents an evolutionary approach to routing protocol analysis for WSNs, which has two unprecedented advantages in that it obtains (i) *quantitative* metrics for the quality of a WSN, and (ii) a large number of faulty WSN configurations, which allow for better generalization of the results.

An evolutionary core generates network topologies, optimizing special heuristic metrics, with the final objective to uncover latent problems in the routing protocol. In the experiments, we focus on the Collection Tree Protocol (CTP) [5] and its TinyOS [9] implementation; we use the TinyOS simulator TOSSIM [10] for the evaluation of a topology. The metrics we consider relate to the number of networking (i.e., radio transmission and reception) *system calls* on WSN nodes, as these form a metric which best correlates to energy depletion (and thus lifetime), independently of the hardware platforms deployed in the WSN. We then uncover a large family of WSN configurations over which CTP causes extremely high traffic, which would drastically diminish the lifetime of a real-world WSN deployment. Based on this, we define a metric, named *degree of disconnection*, which can predict the energy consumption of the protocol over a WSN topology.

The idea of applying evolutionary computation to verification and testing has been explored in several different domains. Preliminary experiments in [16] show that stochastic meta-heuristics are effective in locating the most promising parts of the search space to test complex software modules. A flight system software is verified in [13], where a genetic algorithm outperforms classical random testing techniques. In [4], the operating system of a mobile phone prototype is tested with evolved keyboard inputs, uncovering several power-related bugs.

The remaining of this paper is organized as follows. Section 2 gives a brief overview of system faults in WSNs, including matters of energy consumption. Section 3 describes our approach for detecting anomalous network behaviour in CTP. The experimental results obtained by means of the evolutionary approach are discussed in section 4. Finally, section 5 concludes based on this experience, and outlines future works.

2 Anomalous WSN routing and lifetime

A WSN is a distributed, wirelessly networked, and self-organizing system most often employed for distributed monitoring tasks. Sensing nodes deployed at locations of interest sense, store and send data to one or more *sink* nodes for collection; in turn, a sink may disseminate commands back to the nodes. Network-layer

routing protocols for WSNs most often aim at organizing the nodes into a multi-hop tree-like logical topology for this collection and/or dissemination. Among collection protocols, the Collection Tree Protocol [5] is the *de facto* standard; we analyze its behaviour in this paper.

Given the uncontrolled, distributed nature of a WSN deployment, hardware, software, and networking faults will impact the network's performance; of these, networking anomalies are the most computationally intensive (and thus challenging) to verify against. Field reports [6, 7, 1] list a number of anomalies which unexpectedly impaired deployments, ranging from hardware clock drifts on individual nodes caused by a different response to ambient temperature, to nodes losing routes to the sink due to the network being unpredictably dense for the size of the routing table, and to unforeseen shortness of network lifetime.

An example of such a *lifetime anomaly* is reported in [7]: a link-layer protocol caused most sensor nodes in the deployment to run out of batteries within few weeks. Interestingly, the scientists were never certain *post factum* about the actual cause: *"Back-of-the-envelope calculations reveal that [..] another effect is causing nodes to consume more energy than anticipated. We observed that remote nodes lasted about one week longer. We conjecture that the difference is caused by overhearing less traffic (5 vs. 70 neighbors), but then the small (33%) difference in lifetime indicates that there must be another factor contributing significantly to the nodes' power consumption."* In this paper, we aim to uncover causes for such behavioural effects offline (at WSN design time), before deployment.

When analyzing a network protocol for the purpose of determining network configurations of worst lifetime, a good metric to represent energy depletion on the node is the number of system calls which execute radio receptions or transmissions. It is for radio operations that the battery current draw is highest on any given hardware platform; e.g., on the mainstream MicaZ sensor platform, 19.7 mA are drawn in radio-receive mode, compared to under 8 mA processor draw. In what follows, the radio system calls are our *events* of interest.

3 An evolutionary tool for WSN routing protocol analysis

Uncovering a bug in a complex structure such as a WSN is like finding the proverbial "needle in a haystack": either a bug is discovered, or it is not, with no intermediate possibilities. Such a characteristic significantly impairs any evolutionary algorithm. In fact, the very idea of evolution is based on the accumulation of small differences, each one profitable from the point of view of the organism. In order to tackle such a difficult problem, the evolutionary algorithm is set to design a WSN configuration able to maximize the number of events raised during the simulation. Such a configuration is eventually used to highlight errors and issues. In the following, individual encoding and fitness functions are described.

3.1 Individual description

For CTP verification, an individual represents a candidate configuration of the WSN, encoded as a square matrix $N \times N$, with N the number of nodes in the

network. Each position i, j in the matrix holds the strength of the signal between nodes i and j, expressed in dB. It is interesting to note that, given the nature of wireless transmitters and receivers, the matrix need not be symmetrical.

While a viable connection quality between two nodes can theoretically assume any value above -110 dB (a threshold particular to TOSSIM), we reduce our problem to two subintervals of primary interest:

1. *strong links* with signal strength in the interval $SL = [-30, 0]$ dB, over which small variations may well lead to different behaviours;
2. *weak links* with signal strength below -30 dB, which we equate with no link at all.

We chose the interval SL as such in order to cap the effect that TOSSIM's noise model has upon network behaviour (see subsection 4.1), i.e., to confine the statistical variation among simulations of the same topology, and thus raise the confidence level of our evolutionary experiments (see subsection 4.2). Also, encoding instead the link quality as the integer interval [-31,0] dB might force the evolutionary algorithm to a long exploration before discovering that results with a high number of weak links are interesting; on the other hand, employing an interval like [-60,0] dB might lead to large groups of individuals with different genotype but exactly the same phenotype.

In order to help the algorithm explore the search space more effectively, individuals present two different *alleles* for each *gene* in their genome: a weak link, and a strong link with an integer strength in the interval SL. Since mutation operators can either change a gene to its allele or fine tune the strength, an individual with a weak link in a given matrix position is *close* to an individual with a strong link in the same position regardless of the actual strength of the link. A user-defined occurrence probability is associated to each allele.

3.2 Fitness function

As reported above, evolutionary computation works best when there is a gradual slope towards the best solutions. Thus, if this is not present in the problem, it is crucial to artificially create such a slope in the fitness landscape. With this requirement in mind, two alternative metrics are adopted to identify different kinds of protocol anomalies. Following the intuitive idea that the more a device is stressed, the more likely it is to show possible problems, we maximize, in independent experiments, the following objectives:

1. `maxNetworkEvents`, the maximum number of node-local network system calls (including packet reception and packet transmission) among all nodes;
2. `sumNetworkEvents`, the sum of node-local network system calls (as above) on all the nodes in the network.

With the first fitness function, we aim at finding network configurations where at least one node generates an abnormal number of radio events, and thus will have diminished lifetime. With the latter, we aim at finding network topologies

where a number of the nodes in the network cause a packet storm, which will lead to decreased lifetime for all involved. Based on our experimental results, these two fitness functions are proven smooth enough to distinguish between well-behaved and anomalous configurations.

3.3 μGP and TOSSIM

The algorithm used in the experiments is μGP, an evolutionary framework developed at Politecnico di Torino [14]. Two interesting properties influenced the choice of this particular evolutionary core. First, since μGP's individuals are represented as a succession of macros with a tunable probability of appearance, it is possible to describe an individual structure such as the one presented in subsection 3.1. Second, the design of the framework is based on the notion of an external evaluator, which makes the integration with a WSN simulator quite straightforward.

In order to test the candidate configurations, the open-source simulator TOSSIM [10] is chosen. TOSSIM is a discrete event simulator which allows users to test and analyze TinyOS sensor networks in a controlled and repeatable environment. Besides guaranteeing high fidelity simulations at different levels of granularity (from hardware interrupts to high-level system events), TOSSIM supports two programming interfaces (Python and C++) to control simulations and interact with them (even at runtime). In this work, the powerful scripting and parsing features of the Python interface are used to collect the number and type of different radio events raised by each node.

With reference to Fig. 1, the evolutionary core creates candidate configurations for TOSSIM. All the system calls on each node are logged during the simulation, and parsed successively to create an event hash map. This hash map is finally processed to obtain the fitness function, by simply counting the number of radio events raised by the nodes in the configuration.

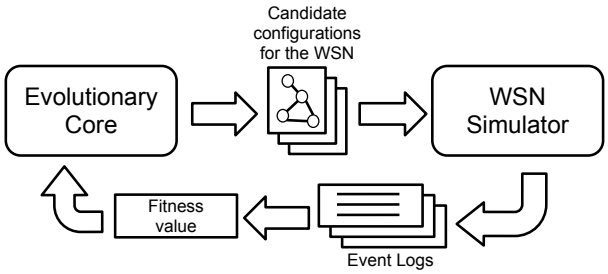

Fig. 1. Conceptual scheme of the proposed evolutionary approach

4 Experimental results

In order to investigate different kinds of protocol anomalies, we ran experiments with three probabilities (25%, 50% and 75%) of having a strong link when gen-

erating a new connection (see subsection 3.1), and the two objectives described in subsection 3.2, namely `maxNetworkEvents` and `sumNetworkEvents`. Thus, six ⟨fitness function, strong link probability⟩ configurations were tested in total.

For simplicity, we consider topologies of 10 nodes booting at simulation time 0; there is enough randomness in the network stack for this setting not to cause network collisions due to synchronicity. Each topology is simulated for 200 seconds. Node 0 is the sink, while the others run CTP to form a multi-hop logical tree topology over the physical graph topology, for the purpose of data collection to the sink. In particular, each node samples one of its on-board sensors every second, and after bundling a number of readings, sends them to the sink.

4.1 Noise analysis

WSNs are affected by a number of stochastic effects, including radio-frequency noise, interference among nodes and from external sources, packet collision, etc. TOSSIM provides an accurate model of the mainstream MicaZ radio stack. In addition to that, to further improve the realism of the simulation (e.g., by taking into account bursts of interference) it is possible to add a statistical *noise model* over the original links' signal strengths. This model is generated automatically from a trace measured experimentally in real-world testbeds, with the generation algorithm based on Closest Pattern Matching [8]. In our experiments, we used the `light-short` noise trace available with TOSSIM.

Several experiments were conducted in order to investigate the influence of the noise on the two fitness functions defined above. More specifically, for each fitness function and percentage of strong links, 200 topologies were sampled randomly in the topology design space, and each one was evaluated 50 times to compute the average fitness value and its variance σ^2. From these experiments it emerged that the variance of the noise dramatically changes depending on the specific topology, making the problem particularly challenging from an evolutionary point of view. Numerical results are presented in subsection 4.3, compared and contrasted with the results obtained with the evolutionary approach.

4.2 Evolutionary process

A detailed exposition of the μGP evolutionary process is out of the scope of this paper. An interested reader may find all relevant information in [14]. μGP was executed with the parameter settings displayed in Table 1, two standard crossover operators (one-point and two-point), and mutation operators that act at the level of a single gene. The evolutionary algorithm was configured in such a way that a $\mu+\lambda$ strategy was used, together with a self-adaptation scheme on the activation of the different genetic operators. Each evolutionary process was allotted a computational budget of 24 hours. It should be noted that the wall clock time for each individual simulation heavily depends on the specific topology and the number of events it generates. Thus, the number of individuals evaluated during each experiment, which in turns affects the number of generations performed, is extremely variable, as shown in Table 2. Additionally, in order to

Table 1. μGP parameters setting

Parameter	Description	Value
`mu`	population size	40
`lambda`	number of genetic operators applied at every step	5
`inertia`	inertia for the self-adapting parameters	0.9
`sigma`	strength of genetic operators	0.9
`tau`	size of the tournament selection	2

Table 2. Configuration of the experiments considered in this study

Fitness function	Strong links	Generations	Evaluations	Best Fitness
`maxNetworkEvents`	25%	206	1231	13554.6
	50%	225	1392	13603.9
	75%	208	1371	12986.1
`sumNetworkEvents`	25%	168	1000	80574.8
	50%	176	1115	87431.1
	75%	221	1406	59995.9

compute a statistically meaningful average fitness value, also in this case each evaluated individual was simulated multiple times with different random seeds. Recalling that the standard error of the mean of a n-dimensional sample whose variance is σ is $\sigma/\sqrt{n}$, and applying the central limit theorem to approximate the sample mean with a normal distribution, it can be proved that a sample size $n = 16\sigma^2/W^2$ guarantees a 95% confidence interval of width W. Thus, in order to guarantee a confidence interval $W = \sigma$ regardless of the actual value of σ (which in any case is not constant in the search space, as we have seen before) we chose $n = 16$ simulations per topology.

4.3 Results: faulty topologies

We comparatively present top individuals obtained by the evolutionary algorithm, and random topologies evaluated separately from the evolutionary experiment; we only detail the experiments configured for 50% strong links, but similar considerations apply to the experiments with 25% and 75% strong links. We first analyze the *top individual* for `sumNetworkEvents`, aiming to derive an explanation for its radio behaviour; Fig. 2 (left) depicts the top individual as an *undirected graph*, computed from the original directed graph so that the undirected edge between nodes m, n exists iff both directed edges $m \to n$, $n \to m$ exist. This shows that the topology is *disconnected* with respect to undirected links; directional links may still exist between graph components, but they prove insufficient for a well-behaved topology. Fig. 2 (right) gives the histogram of radio system calls per node in the network. This shows that not fewer than seven out of ten nodes nearly reach the maximum number of events per node; correlating this fact with the topology of the individual, we can state that a traffic storm is caused among nodes in a component disconnected from the sink.

On the basis of this observation, we define a *metric* for physical WSN topologies, which will qualitatively predict the values of both fitness functions. We call

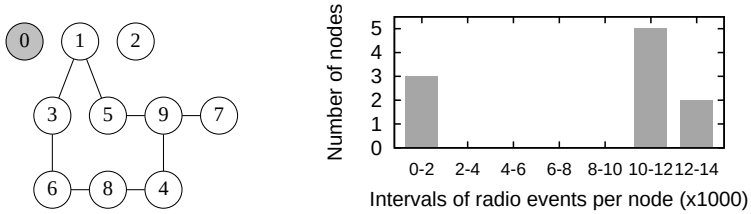

Fig. 2. (left) Undirected physical topology for `sumNetworkEvents` top individual with 50% strong links. (right) Histogram of radio events per node: the top two buckets contain nodes {1, 3, 4, 5, 6, 8, 9}, which form a cycle disconnected from the sink.

this *degree of disconnection*, and we define it over the undirected version of the topology graph G as the number of nodes in all those *graph components* of G which (i) do not include the sink node 0, and (ii) have size greater than 2 nodes:

$$\text{degree of disconnection}(G) := \sum_{\substack{C_i \in \text{ components}(G) \\ 0 \notin C_i,\ \text{size}(C_i) > 2}} \text{size}(C_i)$$

This metric is intuitive: it equals 0 for a *connected* graph G, and every graph component which both does not contain the sink node 0 and is large enough to form cycles contributes to the sum; small, 1- or 2-node components cause few radio events. For a given topology, we conclude that a degree of disconnection strictly greater than 0 is reason for concern with regard to the `maxNetworkEvents` fitness; the higher the value of this metric, the more `sumNetworkEvents` may rise. For the top individual in Fig. 2, this degree of disconnection is 8.

Fig. 3 gives the top 100 individuals and the same number of random topologies for both fitnesses, all with 50% strong links. The fitness value of a topology is shown as the average and standard deviation over 16 simulations (as introduced in subsection 4.2). All top individuals have a degree of disconnection of at least 6 (for the `maxNetworkEvents` fitness) and 7 (for `sumNetworkEvents`). As seen from our random testing, low fitness values correlate well with zero disconnection, which are both marks of well-behaved topologies.

When comparing random tests with our evolutionary algorithm, it becomes clear that the power of the evolutionary analysis is to uncover a large number of faulty topologies, all of higher fitness than found through random testing.

5 Conclusions and future works

Detecting anomalous behaviours in WSNs is a complex task, due to non-trivial interactions between nodes. This paper presented an innovative approach for analyzing the anomalous behaviour of routing protocols in WSNs. Following the intuitive concept that stressing a device is likely to uncover eventual anomalies, an evolutionary framework is set to generate network configurations, with the objective to maximize the number of events raised by the nodes. Experimental results demonstrate that the proposed approach is able to identify network layouts that generate undesired traffic storms, thus potentially being extremely

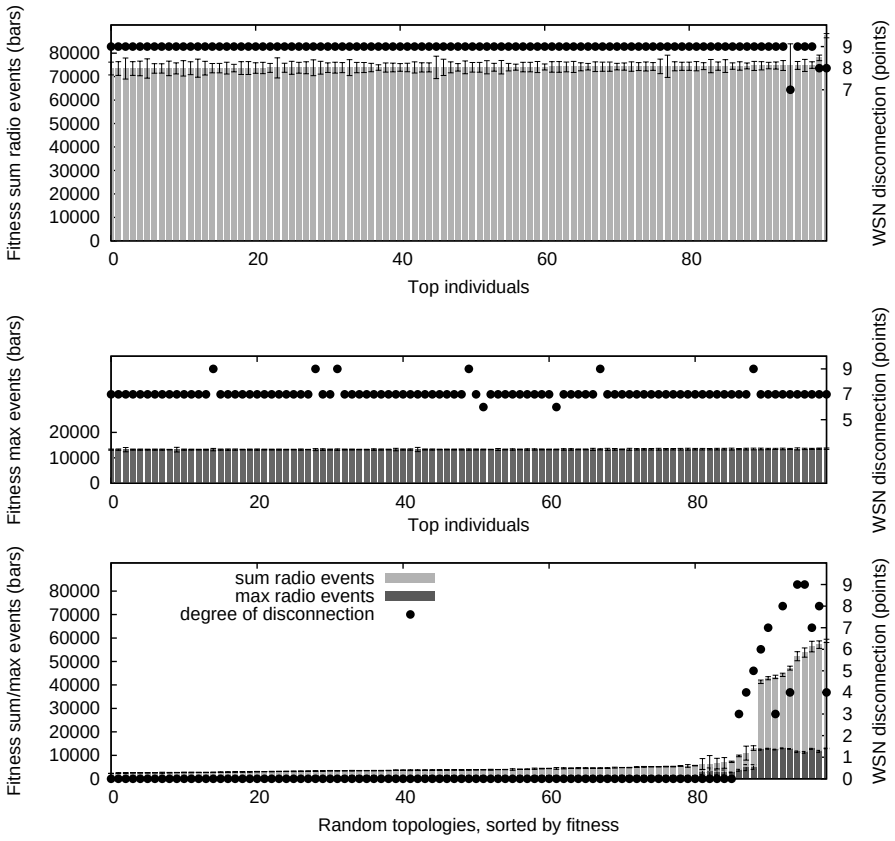

Fig. 3. Overview of fitness values (left y axis) for 100 top and random individuals, all with 50% probability of strong links. Superimposed, we show the degree of disconnection for each individual (right y axis).

useful to analyze and correct lifetime-related issues before network deployment and during the development of new protocols.

Future works will (i) explore the effectiveness of different fitness functions targeting other behavioural faults, e.g., the ratio of delivered data packets, and the number of packets received in duplicate at the sink; (ii) consider the use of a variable-length genome, including node reboots and other node-level events; and (iii) broaden the analysis to include other WSN protocols.

Acknowledgments. INCAS^3 is co-funded by the Province of Drenthe, the Municipality of Assen, the European Fund for Regional Development and the Ministry of Economic Affairs, Peaks in the Delta.

References

1. Barrenetxea, G., Ingelrest, F., Schaefer, G., Vetterli, M.: The hitchhiker's guide to successful wireless sensor network deployments. In: Proc. 6th ACM conference on Embedded network sensor systems. pp. 43–56. SenSys '08, ACM (2008)
2. Bucur, D., Kwiatkowska, M.: On software verification for sensor nodes. Journal of Systems and Software 84(10), 1693 – 1707 (2011)
3. D'Silva, V., Kroening, D., Weissenbacher, G.: A survey of automated techniques for formal software verification. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 27(7), 1165–1178 (2008)
4. Gandini, S., Ruzzarin, W., Sanchez, E., Squillero, G., Tonda, A.: A framework for automated detection of power-related software errors in industrial verification processes. Journal of Electronic Testing 26(6), 689–697 (2010)
5. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P.: Collection tree protocol. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems. pp. 1–14. SenSys '09, ACM, New York, NY, USA (2009)
6. Jurdak, R., Wang, X.R., Obst, O., Valencia, P.: Wireless Sensor Network Anomalies: Diagnosis and Detection Strategies, Intelligent Systems Reference Library, vol. 10, chap. 12, pp. 309–325. Springer, Berlin, Heidelberg (2011)
7. Langendoen, K., Baggio, A., Visser, O.: Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. In: Proc. Int. Conf. on Parallel and distributed processing. pp. 174–181. IEEE Computer Society (2006)
8. Lee, H., Cerpa, A., Levis, P.: Improving wireless simulation through noise modeling. In: Proceedings of the 6th international conference on Information processing in sensor networks. pp. 21–30. IPSN '07, ACM, New York, NY, USA (2007)
9. Levis, P., Gay, D., Handziski, V., Hauer, J.H., Greenstein, B., Turon, M., Hui, J., Klues, K., Sharp, C., Szewczyk, R., Polastre, J., Buonadonna, P., Nachman, L., Tolle, G., Culler, D., Wolisz, A.: T2: A second generation OS for embedded sensor networks. Tech. Rep. TKN-05-007, Technische Universität Berlin (2005)
10. Levis, P., Lee, N., Welsh, M., Culler, D.E.: TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In: Proceedings of the ACM conference on Embedded Networked Sensor Systems (SenSys). pp. 126–137 (2003)
11. Li, P., Regehr, J.: T-Check: bug finding for sensor networks. In: Proceedings of the 9th International Conference on Information Processing in Sensor Networks (IPSN). pp. 174–185. ACM (2010)
12. Mottola, L., Voigt, T., Österlind, F., Eriksson, J., Baresi, L., Ghezzi, C.: Anquiro: Enabling efficient static verification of sensor network software. In: Workshop on Software Engineering for Sensor Network Applications (SESENA) ICSE(2) (2010)
13. Sacco, G., Barltrop, K., Lee, C., Horvath, G., Terrile, R., Lee, S.: Application of genetic algorithm for flight system verification and validation. In: Aerospace conference. pp. 1–7. IEEE (2009)
14. Sanchez, E., Schillaci, M., Squillero, G.: Evolutionary Optimization: the μGP toolkit. Springer Publishing Company, Incorporated, 1st edn. (2011)
15. Sasnauskas, R., Landsiedel, O., Alizai, M.H., Weise, C., Kowalewski, S., Wehrle, K.: KleeNet: discovering insidious interaction bugs in wireless sensor networks before deployment. In: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN). pp. 186–196. ACM (2010)
16. Shyang, W., Lakos, C., Michalewicz, Z., Schellenberg, S.: Experiments in applying evolutionary algorithms to software verification. In: IEEE World Congress on Computational Intelligence (CEC). pp. 3531–3536. IEEE (2008)