Applied Soft Computing 16 (2014) 210–222

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

The impact of topology on energy consumption for collection tree protocols: An experimental assessment through evolutionary computation

Doina Bucur[a,1], Giovanni Iacca[b,*,1], Giovanni Squillero[c,1], Alberto Tonda[d,1]

[a] Johann Bernoulli Institute, University of Groningen, Nijenborgh 9, 9747 AG Groningen, The Netherlands
[b] INCAS[3], Dr. Nassaulaan 9, 9401 HJ Assen, The Netherlands
[c] Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy
[d] INRA UMR 782 GMPA, 1 Avenue Lucien Brétignières, 78850 Thiverval-Grignon, France

ARTICLE INFO

Article history:
Received 3 July 2013
Received in revised form 26 October 2013
Accepted 9 December 2013
Available online 27 December 2013

Keywords:
Collection tree protocol (CTP)
MultiHopLQI (MHLQI)
Wireless sensor networks (WSN)
Evolutionary algorithms (EA)
Routing protocols
Verification
Energy consumption

ABSTRACT

The analysis of worst-case behavior in *wireless sensor networks* is an extremely difficult task, due to the complex interactions that characterize the dynamics of these systems. In this paper, we present a new methodology for analyzing the performance of routing protocols used in such networks. The approach exploits a stochastic optimization technique, specifically an *evolutionary algorithm*, to generate a large, yet tractable, set of critical network topologies; such topologies are then used to infer general considerations on the behaviors under analysis. As a case study, we focused on the energy consumption of two well-known ad hoc routing protocols for sensor networks: the *multi-hop link quality indicator* and the *collection tree protocol*. The evolutionary algorithm started from a set of randomly generated topologies and iteratively enhanced them, maximizing a measure of "how interesting" such topologies are with respect to the analysis. In the second step, starting from the gathered evidence, we were able to define concrete, protocol-independent topological metrics which correlate well with protocols' poor performances. Finally, we discovered a causal relation between the presence of cycles in a disconnected network, and abnormal network traffic. Such creative processes were made possible by the availability of a set of meaningful topology examples. Both the proposed methodology and the specific results presented here – that is, the new topological metrics and the causal explanation – can be fruitfully reused in different contexts, even beyond wireless sensor networks.

1. Introduction

Back in 1620, Sir Francis Bacon steadfastly championed the methodical observation of facts as the means of studying and interpreting phenomena. The original "Baconian method", as described in the *Novum Organum Scientiarum*,[2] has since been replaced in science, yet the importance of collecting and cataloging evidence is not called into question. Oddly enough, after four centuries, a common problem in computer science is precisely the limited availability of "facts" to start formulating new hypothesis from.

The size and the complexity of modern computer systems are skyrocketing, posing serious problems to designers and practitioners. While usually a single component, function or facet may be completely verified, the number of all possible interactions of constituent parts prevents a thorough analysis of the full systems. The problem is further exacerbated whenever the environment must be taken into consideration, since the real world is asynchronous and hardly predictable.

In many practical cases, the existence of some problem, or *bug* if in a software component, is demonstrated by the incorrect functioning of the system. In theory, to conjecture the explanation of an incorrect behavior, one might collect and analyze a set of distinct malfunctioning cases. In practice, however, it may be difficult to pinpoint existing issues: there might be insufficient evidence to faithfully reproduce the scenario, or the triggering cause may be so improbable that pure random simulation would never uncover the fault again.

A paradigmatic example of complex computer systems connected with the environment is represented by *wireless sensor networks* (WNSs). In particular, the analysis of ad hoc WSN's routing protocols is extremely hard, as protocol designers have little to nihil post-deployment information about the occurrence and

* Corresponding author. Tel.: +39 3898088796.
E-mail addresses: d.bucur@rug.nl (D. Bucur), giovanniiacca@incas3.eu, giovanni.iacca@gmail.com (G. Iacca), giovanni.squillero@polito.it (G. Squillero), alberto.tonda@grignon.inra.fr (A. Tonda).
1 All authors contributed equally and their names are listed in alphabetical order.
2 *The New Instrument of Sciences.*

1568-4946/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2013.12.002

cause of malfunctions: "We frequently failed to understand performance results and could not determine who was to blame (i.e., the testbed characteristics, or the routing layer?)" [1]. WSN protocols which performed well in controlled environments had as low as 2% data delivery in the field [2,3]. The observed network lifetimes were also sometimes inexplicably lower than expected: "[the] network dies out: three weeks into the deployment most sensor nodes ran out of batteries. We conjecture that the difference is caused by overhearing less traffic, but [. . .] there must be another factor contributing significantly to the nodes' power consumption" [4]. Yet, both scholars and practitioners openly admit their inability to reproduce faulty scenarios: "it is unclear why collection performs well in controlled situations yet poorly in practice, even at low data rates" [5].

Such a lack of understanding is not surprising. The indoor WSN testbeds used for testing (e.g., MoteLab [6]) form relatively well-connected networks, unlikely to reproduce the type of extreme challenges encountered later in environmental deployments. Furthermore, "experimental results obtained on a single testbed are very difficult to generalize" [1]. Since worst-case scenarios are statistically rare events in the state-space of the problem, non-exhaustive methods, such as testbed analysis [7] or random testing [8], are likely to miss them. On the other hand, using formal verification for analyzing such protocols is computationally prohibitive, and has met with limited success in the past [9–12]. At best, it succeeds in identifying unsafe behavior of a WSN protocol in a few concrete, small-size WSN topologies, which makes it impossible to perform statistics and generalize the cause of the behavior.

This paper addresses the lack of experimental evidence when tackling *lifetime anomalies* in WSNs collection routing, by proposing a general methodology and presenting results for a complex real-world application from the field of ad hoc networking. We exploit an *evolutionary algorithm* (EA) to generate a set of distinct, significant scenarios where the WSN is not behaving correctly. The scenarios are initially generated randomly, and then iteratively refined to potentially trigger critical situations. To let the algorithm evaluate a possible scenario, it is only necessary to define quantitative functions which roughly express "how interesting" that particular scenario is. For example, in our case study, the EA evaluates (i) the overall network radio traffic, and (ii) the maximum node-local radio traffic among all WSN nodes. These two functions measure energy drain for all nodes, and the highest energy drain at a single node, respectively, which are two of the basic definitions for network lifetime used in the literature [13].

In the current contribution, we coupled the meta-heuristic optimization technique and a WSN simulator for generating topologies with abnormally high routing traffic. Then, we demonstrated how these data can be used to discover the cause of the behavior. In more details, we experimented with the *collection tree protocol* (CTP) and the *multi-hop link quality indicator protocol* (MHLQI). For both protocols, we acquired a large set of samples of anomalous WSN lifetime. Using such experimental evidence, we demonstrated that it is possible to establish the *cause* of the misbehavior, and we detected which features of the underlying physical topologies cause a particular protocol logic to trigger unusually high traffic (and thus low lifetime). We call these features *topological metrics*. For each protocol, we first showed the correlation between pairs of ⟨fitness, topological metric⟩ in the samples generated with the EA, then we tested the reverse, i.e., that topologies artificially generated so that they maximize a given topological metric are *sufficient cause* for the protocol to show high traffic.

At the time of writing, no attempt to perform a quantitative causality analysis of worst-case lifetime in WSN routing protocols has been reported in mainstream scientific literature. We find that this lack of knowledge in the field of protocol analysis can be suitably remedied by the application of an EA in a novel, practical methodology, which contributes to the set of real-world applications of EA techniques, as called for in [14,15].

We structure the paper as follows. In Section 2, we give background information on WSN collection routing protocols, evolutionary computation and the specific EA used in the experience. Section 3 describes the method to generate evidence for extreme lifetime in the system, and the method to analyze the evidence and extract a topological correlation and cause. The following three sections present experimental results for both protocols, from obtaining the EA-driven samples of protocol behavior (Section 4), to writing empirical topological metrics and showing their correlation with high network traffic (Section 5), to showing causality by reverse testing (Section 6). Finally, Section 7 draws the conclusions of our work and outlines future developments.

2. Background

2.1. WSN collection routing

A WSN is a distributed, wirelessly networked, and self-organizing system most often employed for distributed monitoring tasks. Powerful and relatively inexpensive, they are widely adopted in many applications, ranging from building surveillance to environmental monitoring [2–4]. In a WSN, each node is a resource-constrained embedded system, and the physical topology of the wireless network typically exhibits heavy link dynamics. Sensing nodes deployed at locations of interest sense, store and forward data to one or more *sink* nodes for collection; in turn, a sink may also disseminate commands back to the nodes.

For successful data collection and dissemination, a WSN often employs an *ad hoc collection routing protocol*. WSN routing protocols are usually designed for datagram-based routing (i.e., they route each data packet independently from any other packet), and are based on adaptive route selection (i.e., they choose a route to forward a packet based on current network traffic conditions, instead of statically). WSN collection routing aims at organizing the nodes, for data collection, into a multi-hop loop-free spanning-tree logical topology rooted in sink nodes. Such protocols are distance-vector protocols which often implement an adaptiveness of the route selection with link dynamics by continuously measuring the quality of a link in a link-quality indicator (LQI), i.e., link costs estimated through link connectivity statistics. This LQI estimation is done on the basis of broadcasted beacon packets (used in all distance-vector protocols by nodes to announce to neighbor nodes both their presence, and their total costs to reach a sink node). Typically, a node broadcasts beacon packets at a fixed interval, and a routing table is maintained by each node on the basis of neighbor information.

All distance-vector protocols are prone to *routing loops*, i.e., inconsistencies in forming a routing tree. Protocols implement various correcting mechanisms for routing loops, such as dropping (instead of forwarding) packets when loops are detected at a node, or speeding up the beaconing interval in a bid to aid tree recovery. Thus, the performance of collection routing depends on the logic implemented by the protocol, on node characteristics (such as radio transmitting power), on the physical topology of the network, and on environmental conditions.

Among the distance-vector collection routing mechanisms applied to wireless sensor networks, MHLQI and CTP are allegedly the most used protocols. MHLQI is an early version of collection routing designed with periodic beaconing (at a fixed rate of 1 beacon per 30 s), and with packet dropping, i.e., a node discards packets when it detects a loop, until a new next hop is found at that node [16]. CTP is the current de-facto standard for WSN collection routing: it builds on the concept of combining the basic distance-vector, beacon-based mechanism with adaptive beacon intervals, designed to broadcast beacons more often in order to reconstruct the routing tree when the network is changing. Sink nodes have a cost of zero; a node's cost is the cost of its next hop plus the cost of their link [5]. Both MHLQI and CTP have been deployed in battery-powered testbeds for environmental, medical or infrastructure monitoring, e.g., [2,17,18].

2.2. *Related performance evaluation studies*

Existing performance studies of such ad hoc protocols have quite low coverage of the state-space of the protocol. For instance, Boukerche [7] simulates protocols on random topologies of 50 and 100 nodes, of which up to 30 packet-injecting nodes. The author empirically learns that two general protocol features cause comparatively high traffic: the use of network-wide flooding, and the use of periodic beacon packets. However, the influence of topological metrics on overhead is not studied, nor is overhead quantified through other than random testing.

In the area of applied model checking based on *qualitative* specifications (i.e., distributed assertions and non-metric temporal logics), Mottola et al. [10] apply formal verification on up to 30 nodes over a data dissemination protocol – the dual of collection. They locate a small set of concrete topologies over which the protocol violates a distributed liveness property. In a similar way, T-Check [9] locates node-local safety and liveness bugs, including some over small static topologies running the CTP implementation in TinyOS.

Regarding *quantitative* analysis, scholars already discovered few interesting properties for WSN collection protocols. For instance, Puccinelli et al. [8] find a protocol-independent metric that correlates with the protocol performance factor. In their case, this performance factor is the ratio of data delivery to the sink node, and the metric is the sum of packet reception rates (gathered experimentally) over all shortest paths from every sensing node to the sink; for this purpose, the shortest paths are calculated post-experiment using Dijkstra's algorithm. They limit the analysis to a small sample of large-scale, well-connected physical topologies. Quite interestingly, such a choice led them to overlook the worst-case scenarios that we found with our methodology.

Concerning the creation of evidence, Baldi et al. presented a seminal work on the use of an evolutionary algorithm for generating a critical test case for an oversimplified TCP/IP network [19]. More recently, Begum et al. proposed a semi-automatic method to generate worst-case data-throughput scenarios for the IEEE 802.11 MAC protocol [20]. Their approach is based on a search algorithm through the state-space of an abstract model in state-machine syntax; this has the disadvantage that the modeling makes some simplifying assumptions for some system features, e.g., the presence of a global clock, and the lack of a capture effect. More importantly, the method only generates worst-case scenarios for a small number of given reference topologies, which (as also in the cases [9,10] above) does not allow for a *generalization* of the causes of the low performance in terms of network context. Finally, the feasibility of coupling an evolutionary algorithm and a WSN simulator for generating a set of critical network topologies was shown [21][3].

2.3. *Evolutionary computation*

Evolution is the biological theory that the origin of all living species is caused by modifications occurring in successive generations. Natural evolution is not a random process: only changes that are beneficial to the individuals are likely to spread into subsequent generations. It is based on random variations, but some are rejected, while others are preserved on the basis of the "fitness" they convey to the individual. Darwin called this principle "natural selection", a quite simple process where randomness "afford materials" [22]. Strikingly, the process only requires to assess the effect of random changes, not the ability to design intelligent modifications.

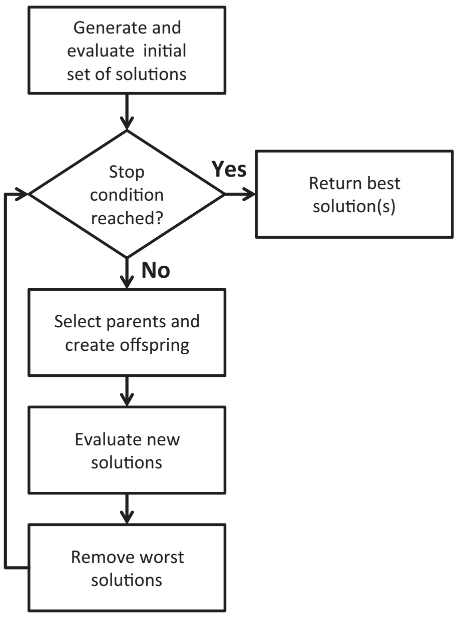

Fig. 1. Flowchart of a generic evolutionary algorithm (EA). Parent selection is usually stochastic, with the best candidate solutions having a higher probability to generate offspring. Removal of individuals is usually deterministic: the solutions are sorted by fitness values, and the worst ones are deleted.

Evolutionary computation (EC) is the offshoot of computer science focusing on algorithms loosely inspired by the theory of evolution – the "evolutionary algorithms" [23]. The definition is deliberately vague since the boundaries of the field are not, and cannot be, sharply defined. EC is a branch of *computational intelligence*, and it is also included into the broad framework of *bio-inspired heuristics*. As noted before, EAs may provide an effective methodology for trying random modifications where no preconceived idea about the optimal solution is required.

Most of the jargon of evolutionary computation mimics the precise terminology of biology. In an EA, a single candidate solution is termed *individual*; the set of all candidate solutions that exist at a particular step of the algorithm is called *population*. Evolution proceeds through discrete steps called *generations*. In each of them, the population is first expanded and then collapsed, mimicking the processes of breeding and struggling for survival (Fig. 1).[4]

[3] This contribution extendeds [21] and includes novel points:

- The scope of the study was broadened adding a second mainstream protocol (MHLQI)
- The maximum size of the networks was increased by five times.
- The set of relevant topological metrics for the energy behavior of both protocols was extended by five times.
- A brand new experimental campaign was added.
- The analysis of correlations in the experimental data was extended.
- A causality link between certain topological features of a network and the energy behavior of certain protocol features was experimentally demonstrated. This result may be effectively used by network engineers in the process of evaluating a protocol design.

[4] It should be noted that some evolutionary algorithms do not store a collection of distinct individuals, but rather evolution is depicted through the variation of the statistical parameters that describe the population, see e.g. the algorithms belonging to the class of *compact optimization* [24], or the "Selfish Gene" algorithm [25,26]. Other

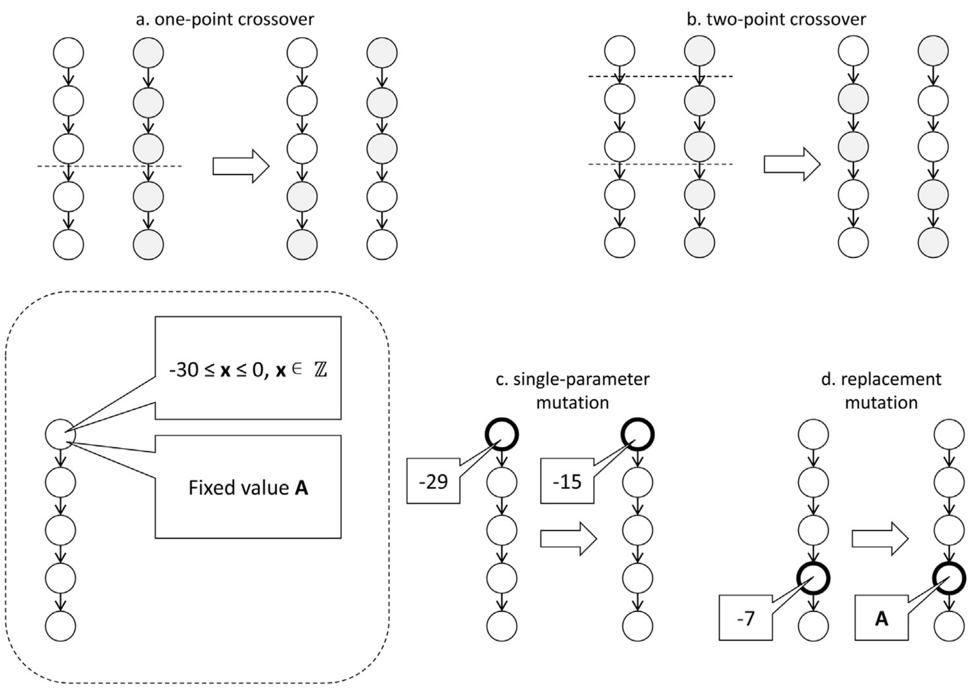

Fig. 2. μGP genetic operators selected for the experiments. The single-point (a) and two-point (b) crossovers produce children individuals by mixing the genome of two parents. In the chosen individual structure, every *locus* in the genome can host two different types of parameters, an integer in the interval [−30,0] and a fixed value **A**. Thus, two different mutation operators are used: the single-parameter mutation (c) changes the value of an integer parameter; while the replacement mutation (d) can switch a *locus* from an integer to a fixed parameter, and vice versa.

The ability of an individual to solve the target problem is measured by the *fitness function*, which influences the likelihood of a solution to propagate its characteristics to the next generation. In some approaches individuals may die of old age, while in other they remain in the population until replaced by fitter ones.

The word *genome* denotes the whole genetic material of the organism, although its actual implementation strongly differs from one approach to another. The *gene* is the functional unit of inheritance, or, operatively, the smallest fragment of the genome that may be modified in the evolution process. Genes are positioned in the genome at specific positions called *loci*. The alternative genes that may occur at a given locus are called *alleles*.

To generate the offspring, EAs implement both sexual and asexual reproduction. The former is named *recombination*; it involves two or more participants, and implies the possibility for the offspring to inherit different characteristics from different parents. When recombination is achieved through an exchange of genetic material between the parents, it often takes the name of *crossover*. Asexual reproduction may be named *replication*, to indicate that a copy of an individual is created, or, more commonly, *mutation*, to stress that the copy is not exact. All operators exploited during reproduction can be cumulatively called *evolutionary operators*, or *genetic operators* because they act at the genotypical level.

An EA outperforms a pure random approach and, at the same time, is more robust than pure hill climbing and other local search mechanisms. Considering their intuitive structure, EAs are quite simple to set up, and require no human intervention when running. Finally, it's easy to trade-off between computational resources and the quality of the results, simply by defining a proper stop condition for the algorithm.

algorithms mimic the evolution of competition between species [27] or cooperation between individuals [28], for problems with specific characteristic [29,30].

2.4. μGP

The EA used in the experiments is μGP, a general-purpose evolutionary toolkit developed at Politecnico di Torino [31,32]. μGP internally represents an individual as a multi-graph, where each node roughly corresponds to a locus of the genome. It is interesting to notice that, differently from most EAs, loci can be occupied by alleles with different characteristics, e.g. integer, float or fixed values, and the probability of appearance of each allele can be tuned. This feature will be extensively used in the following. In μGP, parameter μ indicates the size of the population; λ the number of genetic operators applied at each step (controlling, indirectly, the offspring size); τ the number of individuals in the *tournament selection* [33] used to select the parent individuals; and σ the initial strength of the genetic operators, tweaking the similarity between parents and offspring.

An additional parameter, called *inertia*, controls the self-adapting mechanisms. Self-adapting in EAs is used to slowly shift the focus of the algorithm between exploration and exploitation, depending on the current and past state of the population, improving efficiency and quality of the results [34]. In μGP, in particular, *inertia* regulates the activation probabilities of the genetic

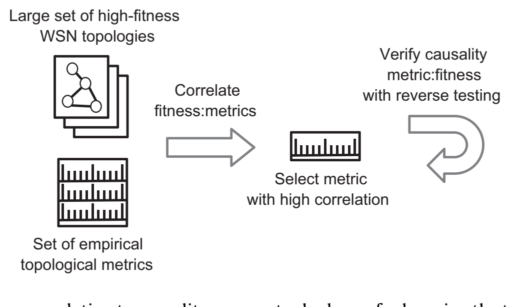
Large set of high-fitness WSN topologies
Correlate fitness:metrics
Verify causality metric:fitness with reverse testing
Select metric with high correlation
Set of empirical topological metrics

Fig. 3. From correlation to causality: conceptual scheme for learning the topological causes for high-fitness WSNs.

Table 1
Parameters for the EA, used in all the experiments. Note that the activation probabilities for all operators and the value of σ, regulating the strength of the mutations, are constantly modified by the self-adapting mechanism during the runs.

Parameter	Value
μ	40
λ	5
τ	2
σ	0.9
Inertia	0.9

Operator	Activation probability
Two-point crossover	0.25
One-point crossover	0.25
Single-parameter mutation	0.25
Replacement mutation	0.25

operators, rewarding the most effective ones; and influences σ, usually reducing the difference between parent and offspring individuals during exploitation.

µGP can use a great variety of genetic operators, depending on the specific characteristics of the individuals' genome. In the following experiments, 4 operators are used: a single-point crossover, a two-point crossover, and two mutation operators. Their behavior, with specific reference to the case study, is summarized in Figure 2 (see Section 3.1 for a detailed description of the individual encoding and Table 1 for the µGP parameter setting).

3. Proposed methodology

We propose to exploit an EA to generate a set of distinct, significant scenarios where the WSN does not behave correctly. Such scenarios are initially generated randomly, then iteratively refined ("evolved") in the attempt of worsening their behavior. Then, the scenarios are analyzed, and one or more metric with high correlation with the network behavior are selected. Eventually, and hopefully, at this point some causal relation may be discovered (Fig. 3).

As stated before, the meta-heuristic optimizer is not used to *demonstrate the incorrectness* of the system, as some erroneous behavior has already been recorded. Nor is it used to minimize the network performance and lifetime in a single case. On the contrary, the EA affords materials for a systematic study, providing the researchers with a conspicuous set of *significant examples*.

To generate the scenarios, the user is requested to provide one or more qualitative fitness functions related to the target problem, i.e. a rough measure of the *attractiveness* of the facts. Such functions do not need to be exact, since they are used as a mere proxy to direct exploration towards the most interesting regions of the state-space of the system at hand.

It is also important to stress that, unlike most applications of evolutionary optimizers, in this study the final fitness value is not so relevant, while it is of capital importance that generated scenarios are different one from another: several examples of moderate malfunctioning are far more useful in an analysis than a single disastrous case. Indeed, the generated set will probably include spurious scenarios, but it will nevertheless provide a starting point far more significant than random sampling, and far more tractable than an exhaustive enumeration.

A conceptual scheme of the evolutionary system adopted to test the WSN behavior is depicted in Fig. 4. For both MHLQI and CTP, we analyze the corresponding implementation in TinyOS [35], the standard operating system designed for WSN nodes. To give µGP an adequate degree of control upon network topologies, we employ configurable TOSSIM [36] simulations of each routing protocol,

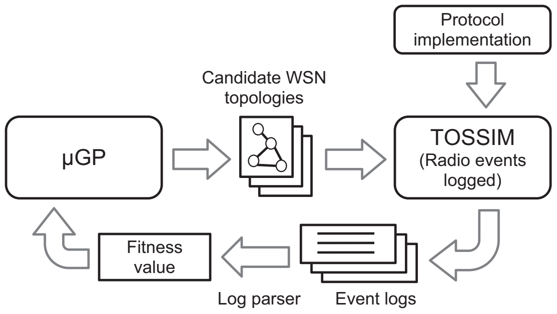

Fig. 4. Conceptual scheme of the proposed approach to the evolutionary-driven simulation of WSN routing protocols.

over a standard full-power data-link protocol. TOSSIM is a real-code simulator, which enables us to analyze the complete software implementation of the protocol, instead of having to remodel the protocol logic into a different language – which is an error-prone task.

Aiming at creating examples of excessive power consumption, we adopt two fitness functions (the *overall network traffic* and the *maximum node-local traffic* among network nodes) as proxy for evaluating the attractiveness of a scenario; these factors directly dictate network and node energy consumption, and thus lifetime. In other words, we quantify energy consumption for sensor nodes indirectly, and in a platform-independent manner: knowing that radio transmission and reception have the highest battery current draw, we estimate energy consumption by simply counting *radio system calls* at each node (see Section 3.2).

As shown in Fig. 4, the evolutionary core of µGP is responsible for creating candidate network configurations, while TOSSIM simulates each and logs all radio system calls (i.e., transmissions and receptions of routing protocol beacons and periodic data packets) generated on each node. These logs are parsed to create an event hash map, which in turn is processed in order to count the radio events raised by the nodes, and obtain the fitness function.

For experimentation, for each evaluated topology we configure all nodes to boot at simulation time 0. Node 0 is the sink in all experiments; at network level, all nodes run either MHLQI or CTP collection routing, while an application-level logic makes all nodes inject data packets in the network periodically, at 5-s intervals. Every single simulation is allowed a simulation runtime of 200 s, during which all radio system calls are logged for post-processing and fitness evaluation; this simulation runtime is sufficiently long to allow either protocol to build a logical routing tree (if one can be built over the given physical topology).

In order to handle the overall randomness of simulations over a network configuration (which then produces a noisy component of both MAX and SUM fitness functions), for every topology evaluation in each of our experiments we execute $n = 16$ simulations, and then average the value of the fitness function obtained over these repetitions. This number of repetitions guarantees a 95% confidence interval of width σ, where σ denotes the standard deviation of the fitness function over an n-dimensional sample of repeated simulations.

3.1. Encoding of evolutionary individuals

For our purposes, an individual is a WSN topology represented as a directed graph. A graph may have various encodings; we choose an $N \times N$ matrix, with N the network size. Each position $\{i, j\}$ in the matrix encodes the signal *gain* from node i to j, and is measured in dB. Diagonal elements (which model node self-connectivity) are not part of the encoding. We remark that, given the directional asymmetry of wireless transmission, we encode topologies as asymmetric matrices, to preserve generality. The long-term asymmetry of communication links is motivated in, for example, [37], which found that directional links "can result from factors such as heterogeneity of receiver and transmitter hardware (leading to differing transmission ranges), power control algorithms (in which nodes vary their transmission power based on their current energy reserves), or topology control algorithms (aimed at reducing interference in the network by computing the lowest transmit power that each node needs to stay connected to the network)."

In TOSSIM, a viable directional link has a floating-point gain greater than -110 dB. Besides gain, link qualities are affected by stochastic effects (i.e., external radio-frequency *noise*, packet collision, and a *capture effect* which allows only the strongest of two signals to be received). These effects introduce a dynamic, noisy component difficult to filter out. These effects are implemented in a TOSSIM simulation, both through an accurate model of the CC2420 radio stack common to many hardware platforms, and by adding a statistical *noise model* to the received signal strength – with this noise model generated from a noise trace [38].

To bound the state-space of the individual encoding, we discretize the problem. First, we only work with integer gain values, and further limit their range by having:

1 *strong links*, encoded with a set of 30 equidistant values above -110 dB; strong links are then superimposed with a low-noise trace which allows a high signal-to-noise ratio;
2 *weak links*, or non-viable links, encoded with a single gain value below -110 dB.

Thus, in the individual genome, each gene may present one of two alleles: a weak link, represented by a fixed arbitrary constant, and a strong link, represented by an integer. The choices we make have two main consequences: on one hand, the low-strength noise limits the statistical variation among simulations of the same topology, raising the confidence level of the evolutionary experiments; on the other, the individual's encoding shapes the search space so that an individual having a weak link in position $\{i, j\}$ is *close* to an individual having a strong link in the same position, regardless of the link's gain. Eventually, this scheme helps the EA explore the search space more effectively, allowing the mutation operators to either switch the allele for a given gene, or fine tune the gain of a strong link,[5] see Fig. 2.

Second, we do a further discretization by only working with particular *network sizes*, and also particular *network densities*, i.e., the percentages of strong links out of the total number of possible network links. For examples, for 50-node WSNs, we analyze networks densities of 1/16 (i.e., a low-density configuration in which each node has on average just over 3 neighbors), 1/8 (a mid-density), and 1/4 (a high-density configuration where a node has on average over 12 neighbors). In µGP, one can associate a user-defined occurrence probability to each allele: in this study, we use this feature to bias the allele probabilities in the initial population, thus focusing each evolutionary experiment on a specific network density.

3.2. Fitness functions

It is a well-known fact that the fitness function heavily affects the behavior of an EA. In particular, an EA produces the best results when there exists a gradual slope towards the best solutions. An almost completely flat (or, on the contrary, an extremely rugged) fitness function is hard to optimize; on the other hand, if one or more "peaks" of attraction are present in the fitness landscape, an EA will naturally converge towards them.

Bearing in mind these considerations, it is then crucial to define a fitness function characterized by a certain slope in its landscape. Following the intuitive idea that the more a system is stressed, the more likely it is to exhibit faults, we search for WSN configurations where the routing protocol causes an abnormally high number of network radio events. Thus, we consider, in independent experiments, the following objectives:

1 *MAX*, the *maximum traffic count at a node*, calculated as the maximum number of node-local radio system calls (i.e., beacon and data-packet reception and transmission) among all network nodes;
2 *SUM*, the *total network traffic count*, calculated as the added number of node-local radio system calls for all nodes in the network.

Both fitnesses are calculated accurately, using added logging mechanisms into the chip-level network driver implementation in TinyOS. By maximizing the MAX fitness function, we aim at finding WSN topologies where at least one node executes an abnormally high number of radio events, consuming more energy and having diminished lifetime. By maximizing the SUM fitness, we instead aim at finding WSN topologies where multiple nodes in the network cause a traffic storm, which in turn will lead to higher energy consumption and decreased lifetime for all nodes involved. Based on our experimental results (see Section 4), we can conclude that the landscape of these two fitness functions has features which allow to discriminate between well-behaved and high-traffic WSN topologies.

4. Gathering evidence from EA-driven simulation

4.1. Experimental results

Table 2 (for MHLQI) and Table 3 (for CTP) summarize the configurations of all the experiments performed in this study. All the topologies generated are available through a public repository.[6] Table 1 reports the parameters used for the EA µGP. All experiments have been run on 6 Intel Xeon 2.40 GHz cores, on a system with 8 GB RAM, Ubuntu 12.04, and kernel 3.2.0-29 x86_64. It should be noted that the wall clock time for each (repeated) simulation of a given topology heavily depends on the protocol (through the number of radio events it generates), on network size, and network density. Larger WSNs combined with high fitnesses do require a longer wall clock time, and thus allow for fewer evolutionary generations and evaluations per time unit: this is intuitively explained by the added computation load needed to simulate, log and process higher counts of radio events. As a consequence, since for the evolutionary experiments we define the stop condition in terms of maximum allowed runtime (24–72 h), the number of individuals (and generations) evaluated during each experiment is not fixed among experiments.

To provide a reference point for our results, for each experimental configuration we measure through simulation the fitness values of simple, connected, 2D-grid "reference" topologies (a standard topology for basic testing of protocols in simulation). We consider grids of sizes 2×5, 4×5, 5×6, and 5×10, respectively. In all these cases, the sink node is situated in a corner of the grid. We make this choice due to the fact that, for these protocols, we experimentally verified that topologies with sinks on the corners are the worst-cases with respect to traffic, compared all other possible sink

[5] To obtain this effect, the mutation operators are implemented in such a way that mutating a strong link creates a weak link in 50% of the offspring, regardless the original gain value.

[6] https://github.com/doinab/evo-network-verification.

Table 2
MultiHopLQI: configuration of experiments and results. Column (A) shows the fitness values for a connected grid topology which we use as reference. Column (B) gives the results of the EA-driven experiments; we denote by (d) and (c) (disconnected or connected) the type of top topologies obtained. Column (C) is obtained through reverse testing, as described in Section 6; here, the fitness is greyed if is lower than in column (B).

			MultiHopLQI						
			(A) Reference grid topology	(B) Evolutionary-driven simulation				(C) Reverse testing	
Fitness function	Network size	Network density	Best fitness (x1000)	Runtime budget (days)	Generations / individuals	Type of topologies	Best fitness (x1000)	Type of topologies	Best fitness (x1000)
MAX: Maximum traffic count at a node	10	1/4	0.45	1	877 / 5212	(d)	1.07	(d)	0.82
		1/2		1	930 / 5166	(d,c)	1.17	(d)	0.94
	20	1/4	0.69	1	469 / 2627	(d)	1.15	(d)	1.11
		1/2		1	430 / 2559	(c)	1.25	(d)	1.09
	30	1/8	1.00	2	339 / 1937	(d)	1.00	(d)	1.01
		1/4		2	316 / 1885	(d)	1.28	(d)	1.13
		1/2		2	288 / 1841	(c)	1.37	(d)	1.24
	50	1/16	1.04	3	516 / 3227	(d)	0.94	(d)	1.04
		1/8		3	542 / 3103	(d)	1.15	(d)	1.18
		1/4		3	496 / 3022	(c,d)	1.33	(d)	1.26
SUM: Total network traffic count	10	1/4	2.05	1	894 / 5159	(d)	3.62	(d)	2.38
		1/2		1	474 / 2703	(d,c)	3.19	(d)	2.74
	20	1/4	4.83	1	239 / 1410	(d)	5.89	(d)	5.42
		1/2		1	419 / 2525	(d,c)	6.94	(d)	5.87
	30	1/8	9.65	2	294 / 1936	(d)	8.51	(d)	7.50
		1/4		2	163 / 1831	(d)	9.18	(d)	8.65
		1/2		2	148 / 1827	(c)	10.58	(d)	9.61
	50	1/16	16.08	3	494 / 3208	(d)	12.68	(d)	12.64
		1/8		3	490 / 3102	(d)	14.53	(d)	13.86
		1/4		3	494 / 2912	(d,c)	16.49	(d)	15.59

Table 3
CTP: configuration of experiments and results. Column (A) shows the fitness values for a connected grid topology which we use as reference. Column (B) gives the results of the EA-driven experiments; we denote by (d) and (c) (disconnected or connected) the type of top topologies obtained. Column (C) is obtained through reverse testing, as described in Section 6; here, the fitness is greyed if is lower than in column (B).

			Collection Tree Protocol						
			(A) Reference grid topology	(B) Evolutionary-driven simulation				(C) Reverse testing	
Fitness function	Network size	Network density	Best fitness (x1000)	Runtime budget (days)	Generations / individuals	Type of topologies	Best fitness (x1000)	Type of topologies	Best fitness (x1000)
MAX: Maximum traffic count at a node	10	1/4	0.7	1	455 / 2437	(d)	19.1	(d)	13.5
		1/2		1	381 / 2280	(d)	13.8	(d)	13.3
	20	1/4	1.3	1	203 / 1098	(d)	13.9	(d)	15.5
		1/2		1	228 / 1433	(d)	11.3	(d)	13.7
	30	1/8	2.0	2	369 / 2262	(d)	16.7	(d)	15.6
		1/4		2	221 / 1424	(d)	14.2	(d)	16.9
		1/2		2	308 / 1938	(c)	7.2	(d)	13.1
	50	1/16	3.2	3	456 / 2906	(d)	16.1	(d)	15.4
		1/8		3	288 / 1822	(d)	13.8	(d)	13.9
		1/4		3	157 / 946	(d)	10.5	(d)	13.9
SUM: Total network traffic count	10	1/4	2.8	1	225 / 1262	(d)	71.3	(d)	114.5
		1/2		1	149 / 1013	(d)	105.1	(d)	97.6
	20	1/4	8.0	1	198 / 1145	(d)	123.2	(d)	254.6
		1/2		1	220 / 1466	(d)	98.6	(d)	164.2
	30	1/8	14.8	2	309 / 1802	(d)	193.2	(d)	370.5
		1/4		2	228 / 1457	(d)	177.8	(d)	370.2
		1/2		2	263 / 1726	(c)	162.5	(d)	224.0
	50	1/16	33.4	3	407 / 2712	(d)	215.9	(d)	633.6
		1/8		3	258 / 1528	(d)	239.1	(d)	593.2
		1/4		3	135 / 873	(d)	312.7	(d)	435.0

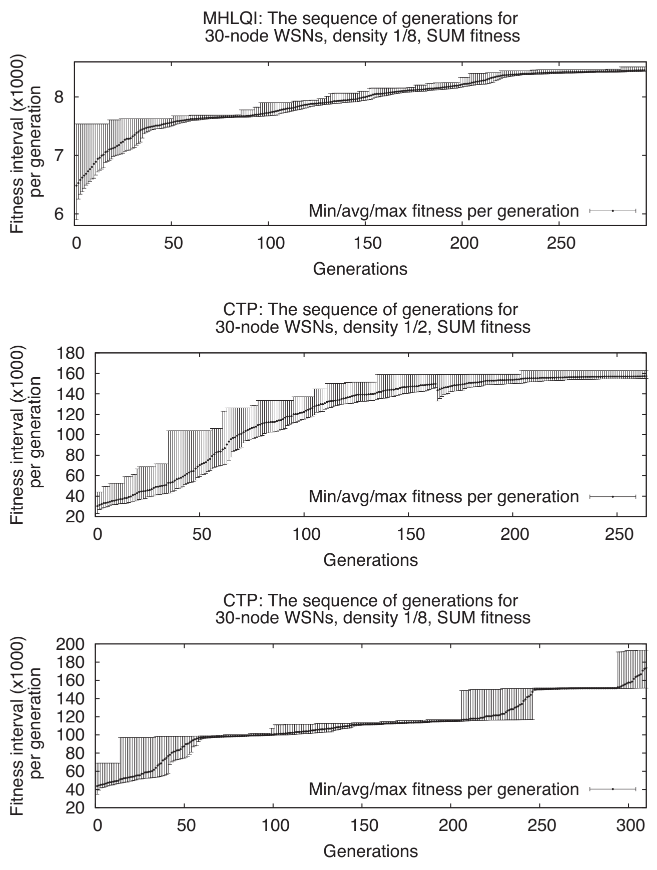

Fig. 5. Fitness trend obtained by EA-driven experiments (top) for MHLQI with (network size: 30, network density 1/8, fitness: SUM), (middle) for CTP with (network size: 30, network density 1/8, fitness: SUM), and (bottom) for CTP with (network size: 30, network density 1/2, fitness: SUM).

placements (such as a centered sink node). We report the value of the fitness functions for such reference grid topologies in column (A) of Tables 2 and 3. In column (B), we show the highest fitnesses obtained by the EA for each configuration. Finally, column (C) shows for each configuration the results obtained with the corresponding reverse testing, as explained in Section 6; the fitness here is greyed if it is not higher than that obtained by the EA in column (B).

From Tables 2 and 3, it becomes clear that (i) MHLQI was not found to exhibit anomalous traffic, while (ii) for all the CTP experiments using the SUM fitness and some of the CTP experiments using the MAX fitness, the best fitness value found by the EA is over an order of magnitude higher than that of the standard, grid reference topology.

Fig. 5 illustrates the sequences of generations created by one MHLQI experiment and two of the CTP experiments. Each generation is shown as the minimum, average and maximum fitness of its individuals. It can be seen that in all cases the evolutionary algorithm is able to constantly improve upon the current solutions along subsequent generations. However, the algorithm behavior (which, in turn, indicates some properties of the fitness landscape) depends heavily on the protocol and the network density. In particular, the difference in the amplitude of the improvement is very large between the protocols: while for MHLQI the total number of network packets increases roughly by 1000 over 250 generations, for CTP this improvement is almost 100 times as high. This provides evidence for a fitness landscape which is extremely different, even for protocols of the same family.

Furthermore, in one case (CTP with network density: 1/2), the algorithm shows a smooth fitness trend which gradually converges

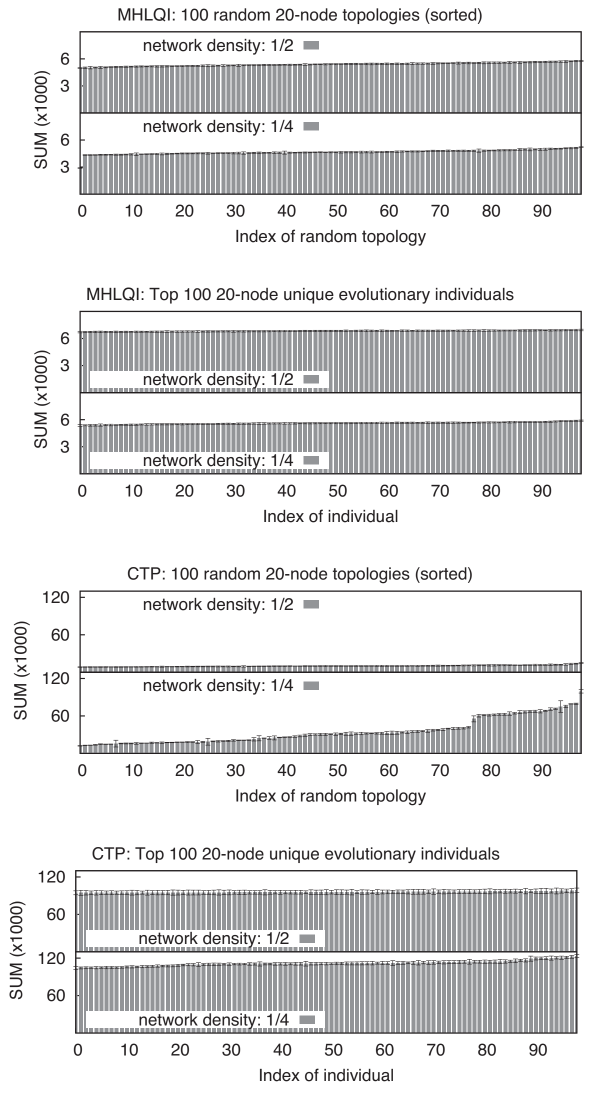

Fig. 6. Fitness evaluations for 100 random topologies and the top 100 unique evolutionary individuals, (network size: 20, network density 1/2, 1/4) for both protocols and the SUM fitness. Every fitness value is plotted as the mean over the 16 fitness evaluations per topology, with the standard deviation over these evaluations shown as an error bar.

to the steady state (as it is also clear looking at the fitness diversity, i.e. the min–max fitness interval which progressively shrinks). In the other CTP experiment shown (network density: 1/8), the fitness trend is instead characterized by abrupt "evolutionary leaps" interspersed with phases where the fitness diversity is low. These behaviors provide evidence for a more complex fitness landscape.

Similar evidence to the difference of fitness landscapes between protocols, and the complexity of the fitness landscape for the CTP protocol can be seen in Fig. 6, which shows the different outcomes in exploring topologies between random and evolutionary experiments. For a given network size of 20 nodes and two network densities (1/2 and 1/4), the figure shows, for each protocol (i) the values of the SUM fitness function for 100 *randomly sampled* topologies, each evaluated over 16 simulations (as also for the EA-driven experimentation), and (ii) the fitness values for the top 100 unique individuals resulted from our EA experiments. Here, we call a topology "unique" in a set if its directional graph is distinct (disregarding the concrete gain value of a strong link) from all others in the set.

On one hand, MHLQI evolutionary experiments never improve significantly over random experiments. On the other, for the particular CTP experimental configuration (network size: 20, network density: 1/2, fitness: MAX) in Fig. 6, the random tests exhibit a seemingly homogeneous field of consistent low fitness values, while the counterpart evolutionary experiment reaches solutions with higher fitness values, and delivers a large set of intermediate solutions of close-to-top fitness. Such WSN configurations are exactly the cases for which random testing is never an effective means for analyzing worst-case protocol behavior.

4.2. Guidelines on EA-driven simulation

In order for WSN practitioners and scientists to reproduce our results and fruitfully use our methodology, a few details of the EA configuration should be borne in mind, namely:

1 *Computational budget*: Defining an optimal runtime for an EA applied to a given verification problem is not a straightforward task. There are several possible stop conditions for an EA, each one offering a trade-off in terms of computational time versus quality of the final result. In our experiments, we assigned a time slot to each experiment. Another possible stop condition is *stagnation* (or *steady state*): if the best individual in the population does not change over a given number of generations, the execution is terminated. A third possible stop criterion is reaching a user-specified value in the fitness of the best individual: for example, if domain knowledge of the problem allows an expert to state a threshold value that highlights a possible fault, the algorithm can be configured to stop once such a fitness value is reached. However, the definition of this threshold for the CTP validation task is impracticable and can only be defined a posteriori.
2 *Local optima*: One problem which may affect the validity of our approach is the presence of local optima in the fitness landscape. This is a serious and well-studied issue for all optimization techniques. Compared to most classic local search methods, such as the Rosenbrock algorithm [39] or the Hooke–Jeeves pattern search [40], whose performance heavily depends on the initial solution (and convergence guaranteed, within some conditions, only towards the closest local optimum), EAs are however more resistant to the attractiveness of local optima: first of all, because of the stochastic element underneath their process; secondly, because they are population-based, and thus sample in a single generation multiple points in the search space. This parallel, scattered sampling of the search space is likely to prevent the algorithm from being trapped into local optima, and helps it converge towards the global one. However, due to the complexity of the search space, the top solutions obtained are not guaranteed to be the global optimum.
3 *Repeatability*: The experiments we reported here are repeatable, although with some limitations. In particular, the sequence of the stochastic operations performed by μGP can be replicated exactly by setting the random number generator's seed in the initialization phase of the algorithm. On the other hand, the remaining stochasticity due to the randomness of the internal logic of CTP and TOSSIM's communication model is difficult to control; to overcome this, it is advisable to simply run multiple simulations for each topology.

5. Inferring quantitative correlations

The sets of top solutions obtained with EA-driven experimentation (Section 4) allow us to conjecture convincingly a set of topological factors which correlate with extreme traffic under collection routing.

To achieve this, we first collected a set of topological features which were hypothetically noted in the literature as likely to have caused a problem with a real-world deployment [4,5]. This set consisted of graph metrics measuring both the overall connectivity of the topology, and the average and maximum node connectivity, for both the directed graph and its translations into undirected graphs. To this set, we added novel topological metrics written particularly for CTP; these metrics were empirically motivated by the topological features of the top individuals found by the EA for CTP. We then computed, for all samples of unique top individuals obtained as evidence with the EA, the correlations between the value of the fitness function and the value of all the topological metrics in our list. For each protocol, we report here the topological metrics which showed highest correlation. As intuitive from the large difference in fitness found by the EA between CTP and MHLQI (Tables 2 and 3), the sets of topological metrics we found to correlate best with high fitness for the two protocols are also different.

In this section, we first motivate empirically the novel topological metrics we defined for CTP, then formally define our best metrics, and finally give the quantitative correlations obtained.

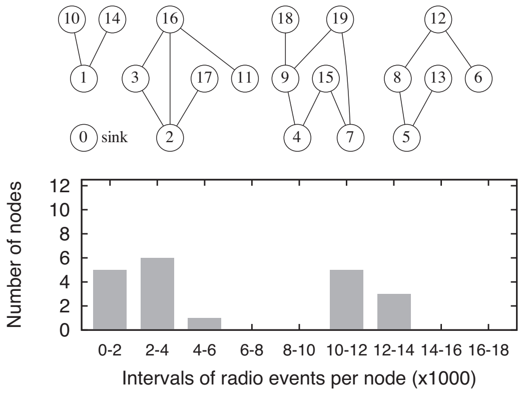

Fig. 7. CTP: top EA individual for (network size: 20, network density: 1/4, fitness: SUM), with resulted best fitness SUM = 123,000. In the histogram of radio events per node, nodes 2, 3, 4, 7, 9, 15, 16, and 19 fall into the top two buckets; the sink node 0 falls into the lowest bucket.

5.1. CTP: empirical motivation for topological metrics

As intuitive motivation for the new topological metrics we found for CTP, we describe in more detail some of the top individuals found experimentally. Figs. 7 and 8 show the top evolutionary individuals resulted from the (network size: 20, network density: 1/4,

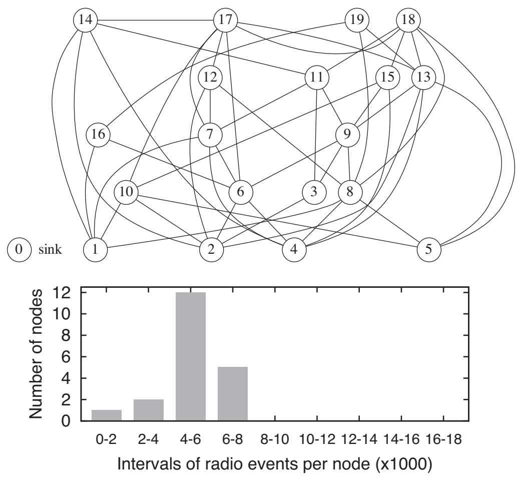

Fig. 8. CTP: top EA individual for (network size: 20, network density: 1/2, fitness: SUM), with SUM = 98,000. All nodes besides the sink have similar, comparably low counts of radio events.

fitness: SUM), and (network size: 20, network density: 1/2, fitness: SUM) experiments, respectively, together with histograms of counts of radio events per node (as resulted from a single simulation). The original, directed, topology is translated here into an undirected graph for clarity; the formal translation for this is given later in Section 5.2. As a note, all nodes in these topologies which are not reachable from the sink node via undirected edges are instead only reachable from the sink via directed edges (not drawn).

From these top individuals given as example, it becomes empirically clear that traffic storms are experienced exactly by nodes located in components *disconnected* from the sink (i.e., components without viable bidirectional paths to the sink), and particularly by those nodes in undirected *cycles*. Furthermore, rather unexpectedly, a single node in a disconnected cycle will experience a comparatively higher traffic count when the cycle is comparatively smaller.

In order to locate what feature of CTP causes this behavior, we contrast here the composition of the total network traffic (i.e., captured by the SUM fitness function) between the 20-node reference grid topology (with fitness value 8,000) and the 20-node top individual from Fig. 7 (with fitness value 123,000). Comparatively, while both have roughly the same network density and inject the same 800 data packets into the network, the reference topology executes only 1,500 beacon transmission and reception events and 6,500 events for data packets. The numbers are 25,000 and 98,000, respectively, for the top individual from Fig. 7. Thus, the ratio of data packets injected to beacon packets rises from 1:2 to over 1:30, and similarly for forwarded packets.

This indicates that the adaptive beaconing mechanism in CTP, which triggers beacon updates in order to resolve apparent routing loops, *fails* to detect (and adapt to) those situations when *no* bidirectional tree can be constructed in the network. This behavior is independent of the network density, and we found it also correlates with another performance factor of collection routing (which is out of the scope of this contribution), i.e., the ratio of data delivery to the sink.

We now formalize our relevant topological metrics, including those measuring disconnection and the presence and number of cycles in a topology.

5.2. Definitions and notation

A WSN topology is represented as a *digraph* $G=(V, E)$ without self-loops, where vertex $0 \in V$ represents the sink node of the collection tree. Given a digraph G, we define the corresponding *undirected graph* $U(G)=(V, U(E))$ so that for any $i \in V$ and $j \in V$, we have $(i, j) \in U(E)$ iff both $(i, j) \in E$ and $(j, i) \in E$. I.e., an edge is preserved in $U(G)$ if both corresponding edges of reciprocal directions are present in G. We denote the number of undirected edges in $U(G)$ by $|U(E)|$.

As usual, for both digraphs and undirected graphs, if a *path* $\langle u, \ldots, v\rangle$ exists between two vertices u and v, then v is said to be *reachable* from u. In particular, we say that a vertex v is *sink-reachable* in the digraph G if v is reachable from vertex 0; the reverse reachability need not hold in G.

The *degree* of a vertex in the undirected graph is the number of edges incident on it; we denote by $\deg(U(G))_{\max}$ the maximum degree among nodes in $U(G)$. The *connected components* of $U(G)$ are the equivalence classes under the reachability relation: all vertices in a connected component are mutually reachable. An undirected graph is said to be *connected* if it has exactly one connected component. Otherwise, the graph is *disconnected* [41]. We call by *sinkless component* a connected component $C=(V_C, E_C)$ of $U(G)$ such that $0 \notin V_C$.

In the undirected graph $U(G)$, a path $\langle v_1, v_2, \ldots, v_k\rangle$ forms a *cycle* if $k \geq 3$ and $v_0 = v_k$. A cycle is called *basic* if the vertices $v_1, v_2, \ldots, v_{k-1}$ are all distinct. The *cycle basis* of $U(G)$ is the set of basic cycles in the graph; any given cycle in the graph can thus be written as a union of cycles in the cycle basis. The *closure* of the cycle basis is the largest union of basic cycles.

Definition 1. (**SU-component**) Given a directed graph G, a connected component $C=(V_C, E_C)$ of the corresponding undirected graph $U(G)$ is called a *sinkless undirected component* (SU-component) if (1) it is sinkless, and (2) any vertex $v \in V_C$ is sink-reachable in G.

Such a sinkless, but sink-reachable undirected component is essentially a connected subgraph of G in which any vertex is reachable from (but cannot itself reach) the sink 0.

Definition 2. (**DD**) For a digraph G, the *degree of disconnection* (DD) is the total number of vertices in SU-components.

Definition 3. (**SU-cycle**) Given a directed graph G, a cycle $\langle v_1, v_2, \ldots, v_k\rangle$ of the corresponding undirected graph $U(G)$ is called a *sinkless undirected cycle* (SU-cycle) if $v_i \neq 0, \forall i = 1, \ldots, k$ (i.e., 0 is not part of the cycle).

Notation 1. (**NSUC**) For a digraph G, we denote by NSUC the total number of vertices on only those SU-cycles which are contained in SU-components. These vertices are reachable from the sink, but cannot themselves reach the sink.

Notation 2. (**CSUC**) We denote by CSUC the number of components containing at least one SU-cycle.

5.3. Correlation coefficients fitness-metrics

We calculate quantitative correlations between the occurrence of high traffic counts and metrics from our set of topological metrics. In Tables 4 (for MHLQI) and 5 (for CTP), we report the *strongest correlations* we found for the two protocols, for all sets of individuals resulted from the various EA-driven experiments. When a particular EA experiment did not yield at least 100 unique topologies of an appropriate type to compute correlations with a particular metric, we mark r with "–"; this occurs, e.g., when too few disconnected topologies were generated for a correlation with the degree of disconnection, DD, and also other metrics pertaining to disconnected topologies. Similarly, we mark r with "–" if the sample does not exhibit sufficient variation in the values of the relevant metric.

These best correlations that we found support the difference found in top fitnesses by the EA-driven experiments. MHLQI shows positive, but *weak* correlations with those topological metrics which are often expected to have an influence on network traffic; for the SUM fitness, these metrics are the density of the undirected topology $|U(E)|$, i.e., the percentage of network links which are bidirectionally strong, and also the degree of disconnection DD; both

Table 4
MHLQI: correlation coefficients r, fitness vs. topological metrics, for the top unique individuals resulted from the EA-driven experiments. When no appropriate sample is available for a calculation, r is marked as "–".

Size	Density	(MHLQI) SUM vs.:		(MHLQI) MAX vs.:	
		DD	$\lvert U(E)\rvert$	DD	$\deg(U(G))_{\max}$
10	1/4	0.71	0.50	0.85	0.71
	1/2	0.71	0.02	0.68	0.20
20	1/4	0.64	0.46	0.55	0.53
	1/2	–	0.69	–	–
30	1/8	0.86	0.83	0.69	0.78
	1/4	0.65	0.41	0.70	0.72
	1/2	–	0.74	–	0.68
50	1/16	0.72	0.77	0.85	0.76
	1/8	0.39	0.75	0.29	0.60
	1/4	–	0.87	0.64	0.43

Table 5
CTP: correlation coefficients *r*, fitness vs. topological metrics, for the top unique individuals resulted from the EA-driven experiments. When no appropriate sample is available for a calculation, *r* is marked as "–".

Size	Density	(CTP) SUM vs.:		(CTP) MAX vs. CSUC
		CSUC	NSUC	
10	1/4	0.94	0.94	0.92
	1/2	0.97	0.91	0.95
20	1/4	0.95	0.91	0.82
	1/2	0.99	0.99	0.73
30	1/8	0.95	0.97	0.97
	1/4	0.94	0.91	0.90
	1/2	–	–	–
50	1/16	–	–	–
	1/8	0.95	0.93	0.86
	1/4	0.75	0.76	0.75

DD and the maximum degree of a node correlate with the MAX fitness.

For CTP, we found not only that the best correlations are with NSUC and CSUC (i.e., the number of nodes in SU-cycles, and the number of components containing such cycles), but also that these correlations are very *strong*. The more disconnected components containing SU-cycles exist in the topology, the higher the MAX fitness of that topology; this is intuitively motivated by the top individual in Fig. 7, where the nodes in the top buckets of the histogram (which form small cycles of sizes 3 and 5) have much higher traffic counts than any of the nodes in Fig. 8 (which form a single, large closure of the cycle basis).

6. Establishing causal relationship

Given the set of topological metrics with best correlations to the two fitness functions (described in Section 5), we aim to verify that, for both protocols, the metrics are a sufficient cause of high fitness. This is particularly important for the case of CTP, where both fitness functions were found by the EA to reach values an order of magnitude higher than the reference topology, and establishing a sufficient cause of this occurrence is crucial.

For this, we designed additional experiments in which we automatically generated, for every tuple (*network size, network density, fitness function*) in our experimental configuration, 100 random test topologies each. These topologies are generated so that they maximize, as much as possible, the topological metrics relevant for that configuration; for example, in the case of CTP, we try to reach high values for both NSUC (whose maximum value is *networksize* − 1), and CSUC. For CTP, we used the empirical generation algorithm

```
Generate empty directed graph G
Add N number of nodes to G

Add to G a random number of bidirectional cycles of
    random size (less than a certain bound B), so that
    the cycles are sink-reachable, disconnected
    from each other, and collectively contain a random
    fraction (above a certain bound F) of the nodes

Add to G a number of random edges, so that the final
    number of edges in G is within a small percentage
    of N*(N-1)*D, avoiding to connect existing components
    when possible, and avoiding to connect any node
    to the sink bidirectionally
```

Fig. 9. Algorithm for empirically generating worst-traffic test topologies of a given size N, and density D, for CTP. Lowering B increases the CSUC metric in the generated topology. We set F to be 90% of N − 1, in order to generate topologies with NSUC within 10% of its absolute maximum value.

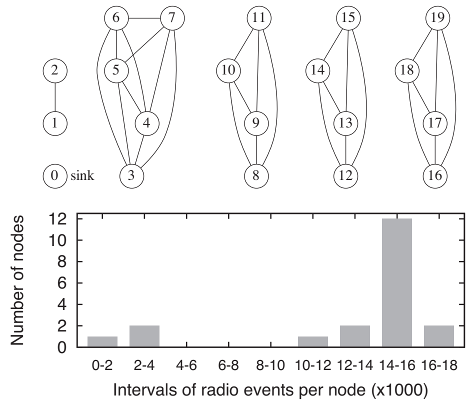

Fig. 10. CTP: top topology resulted from reverse testing, for (network size: 20, network density: 1/4, fitness: SUM), with SUM = 254,478. All nodes except 0,1, and 2 fall in the top buckets with counts of radio events per node of over 10,000; the sink 0 node falls in the lowest bucket.

in Fig. 9. For MHLQI, the generation algorithm is simpler: random graphs are constructed, so that no node can reach the sink, one node has the maximum node degree possible, all edges are bidirectional, and all other edges randomly generated up to the required density.

Tables 2 (for MHLQI) and 3 (for CTP) in Section 4 also present, in column (C), the top fitnesses found through these reverse tests. For MHLQI, this small amount of reverse testing reached high values for the fitness functions, but did not surpass the EA-driven simulation. On the other hand, for CTP, they show that (i) for all the experimental configurations, the reverse testing confirms the results of the evolutionary-driven simulation, and (ii) in 15 out of 20 experiments, the results of the EA-driven tests are surpassed by nearly 200%, as in the case of (network size: 50, network density: 1/16, fitness: SUM). This great increase occurs for the most "difficult" configurations, i.e., those configurations where the fitness landscape is intuitively more complex: experiments pertaining to the SUM fitness function, and large network sizes. We note that this increase in SUM fitness occurred across all network densities. Finally, we try to generalize the behavior of CTP among the various densities analyzed for a given network size. For this, we show in Fig. 12 the amount of traffic experienced by both all reverse-generated topologies (in red) and all EA-generated ones, plotted against one of the

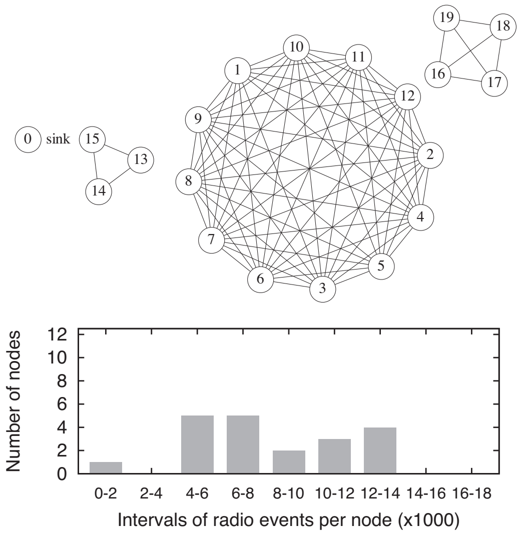

Fig. 11. CTP: top topology resulted from reverse testing, for (network size: 20, network density: 1/2, fitness: SUM), with SUM = 164,239. Nodes 13–19 fall in the top two buckets; the sink 0 node falls in the lowest bucket.

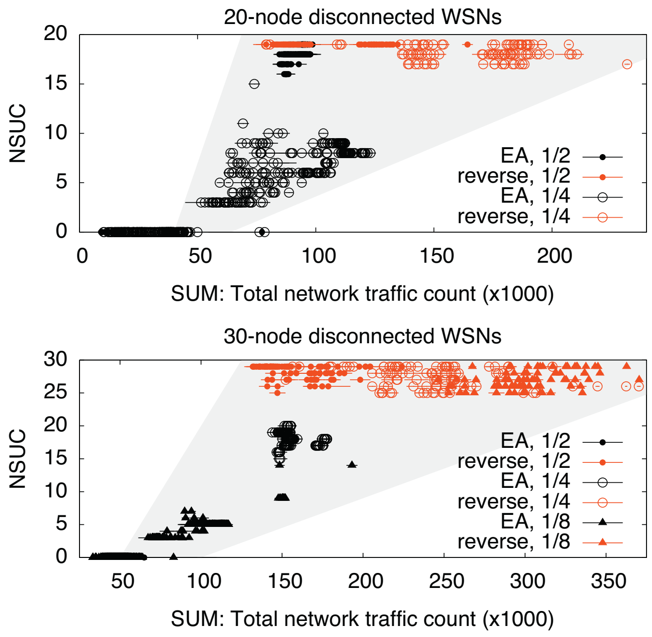

Fig. 12. CTP: scatterplots showing the correlation between the total network traffic (the SUM fitness function) and the NSUC metric. We plot both topologies resulted from the EA-driven simulation (in black), and those resulted from reverse testing (in red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

two relevant topological metrics, NSUC; this shows, qualitatively, the strong correlation between the two across both densities 1/2 and 1/4, with the lowest density showing the highest traffic, and the ability of the reverse testing to generate high-traffic network configurations.

To summarize the large proportion of traffic increase found for CTP, Fig. 13 depicts the ratios between the highest MAX and SUM fitness value found (by either the EA or the reverse testing) and the respective fitness values for the reference topology, across all network sizes and densities. While for MAX, this ratio is under 10 in most of the 10 experimental configurations, a minimum of 10-fold increase in traffic is found for SUM.

To further contrast the top topologies obtained through reverse testing with those obtained with the EA, Figs. 10 and 11 present

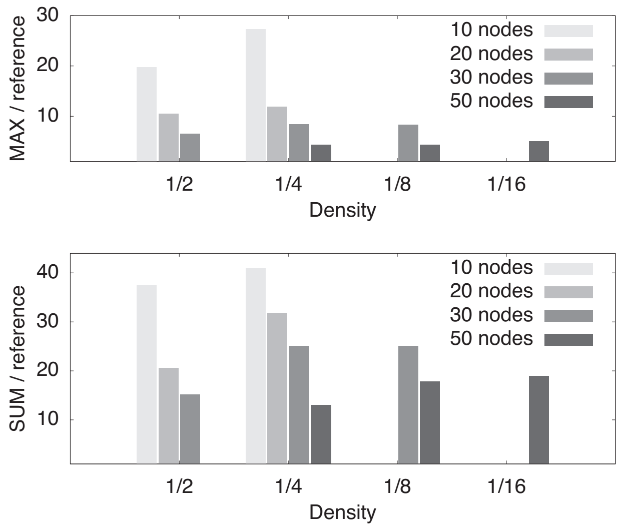

Fig. 13. Ratios of the top fitness values (MAX and SUM) found in this study for CTP, when compared with the respective fitness values for the reference grid topology.

the top CTP topologies from reverse testing resulted from the (network size: 20, network density: 1/4, fitness: SUM), and (network size: 20, network density: 1/2, fitness: SUM) experiments, respectively, together with histograms of counts of radio events per node (as resulted from a single simulation). These should be contrasted with Figs. 7 and 8 obtained originally with the EA-driven experiments. For both densities, the fitness value was nearly doubled through reverse testing using our topology-generation algorithm, with both higher traffic counts for single nodes, and higher number of nodes experiencing high traffic. (It is to note that these new top fitness values obtained may increase further through longer reverse-testing experiments.)

The results from reverse testing confirm that, after having discovered topological causes for high fitness in a particular protocol using EA-driven simulations, it is sufficient to evaluate the particular protocol with this type of efficient reverse random testing.

7. Conclusions

In this paper, we have presented a novel and practical Evolutionary methodology for analyzing worst-cases scenarios of WSN routing protocols. We collected a large set of evolutionarily generated network topologies, from which we deduced heuristic topological metrics correlated with the most problematic cases. In summary:

1 We set up an *evolutionary-driven simulation tool chain* coupling the standard WSN simulator TOSSIM with an EA. This tool chain “evolves” generations of WSN topologies aiming at maximizing two relevant fitness functions. We ran extensive evolutionary experiments and obtained large samples of topologies (of up to 50 nodes and various network densities) over which the protocol logic triggers high fitness values, i.e., high traffic;

2 We empirically extracted predictive *topological metrics* that were shown to correlate with high fitness values in the samples above, and separately verified that the metrics are also a sufficient cause of high fitness. These metrics make our approach *quantitative*, thus comparatively distinct from *qualitative* methods which develop a model of the WSN behavior by analyzing the actual software implementation of the routing protocol, see [9,10].

Here, we focused our analysis on two well-known WSN routing protocols, namely MHLQI and CTP. Although the two protocols belong to the same family, they exhibit extremely different worst-case lifetime. Taking as “reference” point, for both protocols, the amount of traffic triggered over a standard grid topology, we observed that (i) MHLQI, the more simplistic of the protocols, is well-behaved, i.e., does not significantly exceed the reference amount of traffic, while (ii) CTP, the newer and “smarter” protocol, exhibits worst-case traffic one order of magnitude higher than the reference.

In addition to that, we showed that for MHLQI modestly higher traffic is caused by topological metrics which are intuitive and expected, e.g., higher network density and higher node degrees. On the other hand, for CTP, extremely high traffic is caused, less intuitively, by the existence (and number) of physical loops in the network. Finally, we proved that random topologies generated so that they maximize exactly these topological metrics can be used efficiently as *worst-case test scenarios* by protocol designers.

7.1. Future works

The piece of research shown here paves the way to a number of extensions. In our future studies, we will focus our research on various aspects – both methodological and practical – concerning the proposed EA-driven simulation framework and its applications:

1 *Further analysis of wireless sensor networks*: Both our methodology for evolutionary-driven simulation, and the resulting worst-case test generation method, are applicable to other WSN routing protocols and other fitness functions. In our further research, we will show the applicability of this approach to other routing protocols, different fitness functions (e.g., the ratio of data delivery to sink) and networks with dynamic composition.
2 *Extension of the evolutionary framework*: Another possibility to explore will be the application of a domain-specific local search within the evolutionary framework. Incorporating prior domain knowledge into the meta-heuristic, the resulting *memetic algorithm* will likely generate comparatively more interesting scenarios.
3 *Applications beyond WSN*: Although we focused here on the specific problem of generating misbehaving WSN topologies, the proposed EA-driven methodology is applicable to different contexts. In principle, for every optimization problem where a large search space co-exists with the possibility of taking advantage of human expertise, an EA could be effectively used to explore the most interesting areas, obtaining a large set of significant samples that could be later exploited to uncover hidden laws, or derive heuristic considerations on the system under test.

Acknowledgements

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] K. Langendoen, Apples, oranges, and testbeds, in: Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS'06), 2006, pp. 387–396.
[2] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, M. Welsh, Fidelity and yield in a volcano monitoring sensor network, in: Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI'06), USENIX Association, 2006, pp. 381–396.
[3] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, D. Moore, Environmental wireless sensor networks, Proceedings of the IEEE 98 (2010) 1903–1917.
[4] K. Langendoen, A. Baggio, O. Visser, Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture, in: Proceedings of the 20th International Conference on Parallel and Distributed Processing (IPDPS'06), IEEE Computer Society, Washington, DC, 2006, p. 174.
[5] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, P. Levis, Collection tree protocol, in: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09), ACM, New York, NY, 2009, pp. 1–14.
[6] G. Werner-Allen, P. Swieskowski, M. Welsh, MoteLab: a wireless sensor network testbed, in: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN'05), IEEE Press, Piscataway, NJ, 2005.
[7] A. Boukerche, Performance evaluation of routing protocols for ad hoc wireless networks, Mobile Networks and Applications 9 (2004) 333–342.
[8] D. Puccinelli, O. Gnawali, S. Yoon, S. Santini, U. Colesanti, S. Giordano, L. Guibas, The impact of network topology on collection performance, in: Proceedings of the 8th European Conference on Wireless Sensor Networks (EWSN), 2011, pp. 17–32.
[9] P. Li, J. Regehr, T-Check. Bug finding for sensor networks, in: Proceedings of the 9th International Conference on Information Processing in Sensor Networks (IPSN), ACM, 2010, pp. 174–185.
[10] L. Mottola, T. Voigt, F. Österlind, J. Eriksson, L. Baresi, C. Ghezzi, Anquiro: enabling efficient static verification of sensor network software, in: Proceedings of the Workshop on Software Engineering for Sensor Network Applications (SESENA) ICSE(2), 2010.
[11] M. Zheng, J. Sun, Y. Liu, J. Dong, Y. Gu, Towards a model checker for nesc and wireless sensor networks, in: S. Qin, Z. Qiu (Eds.), Formal Methods and Software Engineering, vol. 6991 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 2011, pp. 372–387.
[12] D. Bucur, M. Kwiatkowska, On software verification for sensor nodes, Journal of Systems and Software 84 (2011) 1693–1707.
[13] I. Dietrich, F. Dressler, On the lifetime of wireless sensor networks, ACM Transactions on Sensor Networks 5 (2009) 5:1–5:39.
[14] Z. Michalewicz, Quo vadis, evolutionary computation? On a growing gap between theory and practice, in: Proceedings of the 2012 World Congress conference on Advances in Computational Intelligence (WCCI'12), Springer-Verlag, 2012, pp. 98–121.
[15] Z. Michalewicz, Some thoughts on a gap between theory and practice of evolutionary algorithms, in: IEEE Congress on Evolutionary Computation, 2012, Invited Lecture, http://education.ieee-cis.org/lectures/Invited-Lectures/Some-thoughts-on-a-gap-between-theory-and-practice-of-evolutionary-algorithms
[16] The MultiHopLQI Collection Protocol, 2007 http://www.tinyos.net/tinyos-2.x/tos/lib/net/lqi
[17] J. Ko, T. Gao, A. Terzis, Empirical study of a medical sensor application in an urban emergency department, in: Proceedings of the Fourth International Conference on Body Area Networks (BodyNets'09), ICST, Brussels, Belgium, 2009, 10:1–10:8.
[18] O. Chipara, C. Lu, T.C. Bailey, G.-C. Roman, 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 (SenSys'10), ACM, New York, NY, 2010, pp. 155–168.
[19] M. Baldi, F. Corno, M. Rebaudengo, G. Squillero, GA-based performance analysis of network protocols, in: Proceedings of the Ninth IEEE International Conference on Tools with Artificial Intelligence, 1997, pp. 118–124.
[20] S. Begum, A. Helmy, S. Gupta, Modeling and test generation for worst-case performance evaluation of mac protocols for wireless ad hoc networks, in: IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2009, pp. 1–10.
[21] D. Bucur, G. Iacca, G. Squillero, A. Tonda, An evolutionary framework for routing protocol analysis in wireless sensor networks, in: EvoStar/EvoApplications/EvoCOMNET: 15th European Conference on the Applications of Evolutionary and Bio-inspired Computation, Springer-Verlag, 2013.
[22] C. Darwin, On the Origin of the Species by Means of Natural Selection: Or, The Preservation of Favoured Races in the Struggle for Life, John Murray, London, 1859.
[23] A. Eiben, J. Smith, Introduction to Evolutionary Computing Natural Computing Series, Springer, Berlin, 2003.
[24] F. Neri, G. Iacca, E. Mininno, Compact optimization, in: Handbook of Optimization, Springer, Berlin/Heidelberg, 2013, pp. 337–364.
[25] F. Corno, M.S. Reorda, G. Squillero, A new evolutionary algorithm inspired by the selfish gene theory, in: Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence, IEEE, 1998, pp. 575–580.
[26] F. Corno, M. Sonza Reorda, G. Squillero, Optimizing deceptive functions with the SG-clans algorithm, in: Proceedings of the 1999 Congress on Evolutionary Computation (CEC 99), vol. 3, IEEE, 1999.
[27] P.J. Angeline, J.B. Pollack, Competitive environments evolve better solutions for complex tasks, in: Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, CA, 1993, pp. 264–270.
[28] M.A. Potter, K.A. De Jong, Cooperative coevolution: an architecture for evolving coadapted subcomponents, Evolutionary Computation 8 (2000) 1–29.
[29] P.J. Darwen, X. Yao, Co-evolution in iterated prisoner's dilemma with intermediate levels of cooperation: application to missile defense, International Journal of Computational Intelligence and Applications 2 (2002) 83–107.
[30] A. Tonda, E. Lutton, G. Squillero, A benchmark for cooperative coevolution, Memetic Computing 4 (2012) 263–277.
[31] E. Sanchez, M. Schillaci, G. Squillero, Evolutionary Optimization: the μGP Toolkit, 1st ed, Springer Publishing Company, Incorporated, Berlin, 2011.
[32] G. Squillero, MicroGP – an evolutionary assembly program generator, Genetic Programming and Evolvable Machines 6 (2005) 247–263.
[33] A. Brindle, Genetic Algorithms for Function Optimization, 1981.
[34] K.A. De Jong, An Analysis of the Behavior of a Class of Genetic Adaptive Systems (Ph.D. Thesis), Dept. of Computer and Comm. Sciences, Univ. of Michigan, Ann Arbor, MI, 1975, Univ. Microfilms, 1975.
[35] P. Levis, D. Gay, V. Handziski, J.-H. Hauer, B. Greenstein, M. Turon, J. Hui, K. Klues, C. Sharp, R. Szewczyk, J. Polastre, P. Buonadonna, L. Nachman, G. Tolle, D. Culler, A. Wolisz, T2: a second generation OS for embedded sensor networks, in: Technical Report TKN-05-007, Technische Universität Berlin, 2005.
[36] P. Levis, N. Lee, M. Welsh, D.E. Culler, TOSSIM: accurate and scalable simulation of entire TinyOS applications, in: Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys), 2003, pp. 126–137.
[37] J.G. Jetcheva, D.B. Johnson, Routing characteristics of ad hoc networks with unidirectional links, Ad Hoc Network 4 (2006) 303–325.
[38] H. Lee, A. Cerpa, P. Levis, Improving wireless simulation through noise modeling, in: Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN'07), ACM, New York, NY, 2007, pp. 21–30.
[39] H.H. Rosenbrock, An automatic method for finding the greatest or least value of a function, The Computer Journal 3 (1960) 175–184.
[40] R. Hooke, T.A. Jeeves, Direct search solution of numerical and statistical problems, Journal of the ACM 8 (1961) 212–229.
[41] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 3rd ed., MIT Press, Cambridge, MA, 2009.