Using Evolutionary Dynamic Optimization for Monitor Selection in Highly Dynamic Communication Infrastructures

Robin Mueller-Bady
Martin Kappes
Frankfurt University of Applied Sciences
Frankfurt, Germany
[mueller-bady,kappes]@fb2.fra-uas.de

Inmaculada Medina-Bulo
Francisco Palomo-Lozano
Universidad de Cádiz
Puerto Real, Spain
[inmaculada.medina,francisco.palomo]@uca.es

ABSTRACT

In this paper, we address the problem of applying evolutionary dynamic optimization of network monitoring to highly dynamic communication network infrastructures.

One major challenge of modern communication networks is the increasing volatility due to, e.g., changing availability of nodes and links, load of paths, or attacks. While optimization of those dynamic networks has been an important application area since decades, new developments in the area of network function virtualization and software defined network facilitate a completely new level of automated dynamic network optimization. Especially in mobile networks, changes can be observed to appear swiftly. Thus, using population-based heuristics becomes challenging as reevaluation of all candidate solutions may become time-wise impossible and operations need to rely on possibly obsolete fitness values.

Here, an established method has been applied to solve the dynamic monitor selection problem on multiple real-world problem instances using a different simulated level of change. Statistically significant results of the proposed method have been compared to the performance of a *best-of-multiple selection local search (BMS LS)* heuristic. As the results show, optimization reaches results of high quality even under difficult circumstances.

CCS CONCEPTS

• **Networks** → **Network dynamics**; • **Theory of computation** → **Network optimization**; Evolutionary algorithms; Randomized local search;

ACM Reference Format:
Robin Mueller-Bady, Martin Kappes, Inmaculada Medina-Bulo, and Francisco Palomo-Lozano. 2018. Using Evolutionary Dynamic Optimization for Monitor Selection in Highly Dynamic Communication Infrastructures. In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205651.3208252

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07…$15.00
https://doi.org/10.1145/3205651.3208252

1 INTRODUCTION

Communication network infrastructures have grown massively over the past decades, increasing its quality and quantity. While the amount of communication-enabled devices increased due to the development of mobile communication, internet-of-things, smart homes and a general increase of communication, also the requirement for network quality properties, e.g., bandwidth and latency, increased. Especially the mobile-broadband subscriptions, which have grown over 20% annually over the past 5 years and which are expected to reach a globally total of 4.3 billion soon [11], are a huge challenge for internet service providers due to the mobility of the clients. Handling ordinary events, e.g., roaming or planned downtimes of nodes, is predictable even during the network design phase. As opposed to ordinary events, handling extraordinary events, e.g., overload of nodes due to attacks or hardware failures, must be targeted on demand and can not usually be planned beforehand without spending a huge amount of extra resources for the infrastructure.

Optimization in the area of network infrastructures is an important research area since decades. However, current developments in the area of *Software Defined Networking (SDN)* and *Network Function Virtualization (NFV)*, in conjunction with common hardware virtualization, offer groundbreaking possibilities in automated dynamic network optimization as in, e.g., [18]. While the research focus of network optimization had been on the robust, resilient and efficient design and maintenance of stationary networks and subnetworks, SDN and NFV enable a bird's-eye view on the whole administrative domain and programmatic access of network properties, e.g., traffic shaping, bandwidth management, dynamic routing or monitoring. This facilitates network security applications on a new level by enhancing early detection of threats like botnets, (distributed) denial of service attacks, or propagation of malware [14].

In this paper, we address one of the most basic but also most important steps in securing communication infrastructures: dynamic network monitor selection. As it can be observed that growing communication networks are becoming more dynamic and thus volatile, it is possible that there is a very limited time window between two network changes. Therefore, the simulated model change is applied after each evaluation. Implicitly, it is not possible to re-evaluate all candidate solutions of the evolutionary heuristics' population such that the heuristic relies on obsolete fitness information for selection operations.

The remainder of this paper is structured as follows: In the following Section, the related work and current research is presented, followed by the problem definition in Section 3 and the proposed

solving strategy in Section 4. Section 5 describes the experimental setup for our empirical experiments, while Section 6 discusses the results, finishing with a conclusion and outlook in Section 7.

2 RELATED WORK

Evolutionary Dynamic Optimization (EDO) [1, 27] is an important field in the area of heuristic optimization and has grown over the past decades, as shown by the survey papers of Nguyen et al. [24] and Cruz et al. [7]. Application has been done to several different problem areas, e.g., structural optimization of electrical grounding grids [23] and car distribution systems [19] but also to network optimization, e.g., in wireless sensor network design [25] and to the dynamic shortest path routing problems in mobile ad-hoc networks [29]. Heuristic optimization in static, sometimes referred to as stationary, network problem instances, is a recent topic, e.g., solving topological design problems having multiple objectives [8].

While tweaking the optimization method is not the major focus of this paper, it is still important to chose the, to the best of current knowledge, optimal method for solving the underlying problem. However, different studies disagree which is the best current method for solving the monitor selection problem, i.e., the modified MVC problem. Chauhan et al. have shown that for solving the MVC problem in stationary scale-free networks, an evolutionary approach is generally more efficient than a local search [4], while two of the most efficient applied approaches on massive graphs, NuMVC [3] and FastVC [2], are both based on local search. Thus, using a hybrid approach using an evolutionary algorithm with local search components seems to be a reasonable choice. There also exist exact optimization approaches [5] which have been used to solve the general MVC in static environments. However, the presented approaches on solving the MVC are all applied to static graph topologies, while the problem studied in this paper covers optimization in dynamic network instances.

It can be observed that new methods for solving problems in (dynamic) network problems are often benchmarked using synthetic problem instances. One reason for this is the low amount of real-world problem instances available, as information about network topologies is usually considered as mission- or even safety-critical and is therefore not disclosed. Another reason might be that usually a general applicability of the respective method wants to be achieved over a broad spectrum of different instances. Here, we focus on the application and behavior of the presented method to real-world network models, which reflect the crucial properties of real-world networks [15].

3 THE DYNAMIC MONITOR SELECTION PROBLEM

Monitoring is a first but crucial step towards a secure, robust and resilient network. The common way to capture network traffic is to use monitors on nodes, such that the traffic flowing on all adjacent edges, in the context of networks often referred to as links, can be monitored. In order to reach the goal of monitoring the whole network, each edge in the network must be covered by at least one adjacent monitor.

On the one hand, a robust way to monitor the whole network is to implement monitors on all nodes. As monitoring on a node also implies performance loss, maintenance effort and possible other (monetary) costs, the aim is to use the minimal amount of necessary monitors for covering the whole network. On the other hand, uncovered edges may conceal important information and therefore lead to possible threats. Thus, it is important to find (a) the optimal amount and (b) the optimal positions of monitors in the network in order to cover all edges. For the given problem, it is assumed that a monitor can be applied to each node without further restrictions. However, it is possible to introduce further constraints here as well, e.g., by classifying nodes according to priorities based on their position, performance, cost, node types etc.

Mathematical graphs are a common and natural way to model communication networks. For the remainder of this paper, it is assumed that the model of a communication network can be represented using a finite simple graph, $G = (V, E)$, with V being the set of vertices, representing the nodes, and $E \subseteq V \times V$ being the set of edges representing the connecting links between nodes, as defined in [22]. The described problem is closely related to the well known minimum vertex cover problem as described by Karp [12]. It is defined as a subset of vertices $V' \subseteq V$ such that for each edge $(v_i, v_j) \in E$, either $v_i \in V'$ or $v_j \in V'$, or both. The MVC is a vertex cover where $|V'|$ is minimal for all possible vertex covers of network model G. As the underlying monitor selection problem can be generalized to the described MVC problem, it can be shown that it is also a $\mathcal{NP}$-hard optimization problem. For brevity reasons, the proof is omitted here in this paper.

In addition to the common MVC problem described before, the dynamic monitor selection problem uses an edge weighting function indicating the monitoring priorities of edges, such that the network model definition is extended to $G = (V, E, w)$ with $w : E \rightarrow \mathbb{N}$ being a non-negative weighting function. Furthermore, the dynamic monitor selection problem must be solved on a dynamically changing network model. Within a given optimization period, $[t_{start}, t_{end}]$, the optimization has to react on changes of the network model providing new optimal solutions. Thus, the quality of a solution depends on the covered edges, the amount of monitors used and the time point at which the solution was acquired. Details on the optimization method will be given in Section 4.

4 PROPOSED SOLVING STRATEGY

The proposed solving strategy for the dynamic monitor selection problem is a hybridization of a robust evolutionary algorithm for exploration and a swiftly converging local search for exploitation of the search space. This hybrid heuristic, *Local Search Evolutionary Algorithm (LS EA)*, is an established method and has been applied for solving the monitor selection problem in occasionally changing environments [21]. In the following, the individual components of the proposed solving strategy are described in necessary detail.

4.1 Evolutionary Algorithm

The first component is the evolutionary algorithm based on a common generational EA. Initially, a random population of candidate solutions (individuals) is created and evaluated. Then, two distinct parents are selected from the population using a specified selection operator. Those parents are recombined using a recombination operation creating two children as new individuals, which are then

modified using a given mutation operation. Finally, the population containing all former individuals and created children, are reduced to its defined initial size using selective pressure induced by a selection operation. The process is repeated until a given termination condition is satisfied, e.g., until a maximum number of evaluations or a certain quality threshold is reached.

There are different formats to represent individuals in EAs for graph-based problems, as shown by Gen et al. [10] and Doerr et al. [9]. For the dynamic monitor selection problem, it seems reasonable to follow a vertex-based approach as follows. Let $\bar{x} = (x_1, \ldots, x_n)$ be a binary n-tuple representing a candidate solution, such that each x_k resolves to whether a monitor is present on the corresponding vertex v_k in the graph or not:

$$x_k = \begin{cases} 1, & \text{if } v_k \text{ is selected as a monitor} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Having an individual as previously defined, the number of monitors is the first indicator for the quality of the solution:

$$monitors(\bar{x}) = \sum_{k=1}^{n} x_k \tag{2}$$

where a lower amount of monitors indicates a higher quality.

However, individuals may represent infeasible solutions, i.e., there exist one or multiple edges in the graph not being covered. Thus, the solution is penalized using a linear distance function [6] based on the given weighting w of the edge:

$$penalty(\bar{x}) = \sum_{\{v_i, v_j\} \in S} w(v_i, v_j) \tag{3}$$

where $S = \{\{v_i, v_j\} \in E \mid x_i = 0 \wedge x_j = 0\}$. Finally, the result is aggregated into a scalarized fitness value:

$$f(\bar{x}) = monitors(\bar{x}) + penalty(\bar{x}) \tag{4}$$

The goal of this minimization problem is to find a candidate solution $\bar{x}_0 \in S$, such that $f(\bar{x}_0) \leq f(\bar{x}), \forall \bar{x} \in \{0,1\}^n$.

As evaluation of candidate solutions is done in a dynamic environment, the fitness function including its individual components are time-dependent in the given optimization period $[t_{start}, t_{end}]$:

$$f_t(\bar{x}) = monitors_t(\bar{x}) + penalty_t(\bar{x}) \tag{5}$$

In this specific case, the optimization period ranges from 0 to the maximum number of allowed evaluations. In order to avoid the problem of having a variable-length candidate solution encoding, differences of models are relative to a given base model, setting particular nodes and edges active or inactive. Therefore, both, $monitors_t(\bar{x})$ and $penalty_t(\bar{x})$ are time-dependent, as both underlying sets may differ in the amount of active elements for each consecutive time point t.

4.2 Best-of-Multiple Selection Local Search

Local search is an optimization heuristic based on searching neighboring solutions of a given candidate solutions and a neighborhood relation. Usually, the search is iteratively performed on all neighbors until no better neighbor can be found while continuously replacing the currently searched candidate solution with the improved one. Here, the neighbors are determined using a Hamming distance of 1 as neighborhood relation, i.e., the difference in the encoding of two candidate solutions is exactly 1. However, as an exhaustive search of all neighbors is inefficient for large problem instances, the number of searched neighbors is restricted by the k-value, according to a *Best-of-Multiple Selection (BMS)* as proposed by Cai [2]. The original BMS has been adapted to fit the requirements of the dynamic monitor selection problem, as indicated by the local search algorithm as depicted in Algorithm 1, forming the BMS local search (BMS LS).

Algorithm 1 Scheme of the local search method

```
Require: Initial individual ind, parameter k, comparison function f
  function LocalSearch(ind, k, f)
    currentOptimum ← ind
    repeat
      neighbors ← GET_K_RANDOM_NEIGHBORS(currentOptimum, k)
      bestNeighbor ← OPTIMUM(neighbors)
      if f(bestNeighbor) > f(currentOptimum) then
        currentOptimum ← bestNeighbor
      end if
    until bestNeighbor ≠ currentOptimum
    return currentOptimum
  end function
```

4.3 LS EA

Finally, both the evolutionary algorithm and the local search are combined forming the hybrid LS EA used for optimization of the dynamic monitor selection problem. First, the local search is applied to the current found optimum value. Then, the evolutionary operators are applied on the population including the optimum found by the local search. The optimization is repeated until the termination condition is satisfied, i.e., the maximum allowed number of evaluations is reached. The scheme of the algorithm is depicted in Algorithm 2. The used parameters are described in Section 5.

Algorithm 2 Hybrid local search EA (*LS EA*)

```
population ← INITIALIZE()
EVALUATE(population)
while termination condition not satisfied do
  currentOptimum ← REMOVE_CURRENT_OPTIMUM(population)
  bestNeighbor ← LOCALSEARCH(currentOptimum)
  population ← population ∪ bestNeighbor
  SELECT_PARENTS()
  RECOMBINE_PARENTS()
  MUTATE_OFFSPRING()
  EVALUATE_OFFSPRING()
  SELECT_INDIVIDUALS()
end while
```

5 EXPERIMENTS

In this Section, experiments and experimental setup are described including problem instances and parameters for the proposed solving strategy.

5.1 Experimental Setup

For the optimization, the LS EA as described in Section 4 is used. Parameters of the EA are shown in Table 1.

Generally, as the proposed solving strategy generally follows a generational EA approach. For each experiment, the number of evaluations is restricted to 100 000. Results of 100 experiment repetitions for each parameter configuration and problem instance have been obtained in order to gain statistical significance. The population size is set to 10 as short iterations are required in order to be able to react swiftly on events in the network. It is observable that a sufficiently large amount of genetic diversity is already achieved due to the rapid changes of the underlying model, making a higher population size irrelevant.

In each EA iteration, 10 children are created using uniform crossover and bit-flip mutation with a mutation rate of 5%. Selection of parents and survivors is both done using tournament selection having a tournament size of 5.

For the BMS LS part of the optimization process, a k-value of $k = 50$ is used, i.e., 50 neighbors of the current solution are searched for an improved candidate solution. The current best individual is picked for participation in the local search optimization. As all individuals except the last evaluated one are likely to have obsolete fitness values, it is possible that another individual from the generation may have a higher fitness value. This is, however, not considered for the optimization due to the strict time constraints.

For the experiments conducted, the change of the underlying network model is introduced after each evaluation. In order to simulate different activity levels in the network, the change of the network is divided into different change levels from a base model: 5%, 10%, and 25%. Having a change level of 5%, implies that arbitrary 5% nodes from the given base model are switched to inactive and do neither participate in the network communication nor count for the fitness evaluation. Edges adjacent to inactive nodes are, per definition, also set inactive having the same implication as inactive nodes. Having two consecutive changes using a change level of 5% implies that a maximum difference of 10% of all nodes and adjacent edges is possible. Weightings of active edges are preserved.

The results are obtained using the parallelization framework *multijob* [20] and GNU parallel [28] on a cluster configuration with two servers. The first server is composed of 2x Intel Xeon E5-2690 v4 @ 2.60GHz CPUs (28 cores, 56 hyperthreads) and 126 GB RAM and the second of 2x Intel Xeon CPU E5-2650 v3 @ 2.30GHz CPUs (20 cores, 40 hyperthreads) and 62 GB RAM, both using a Debian Linux as operating system.

5.2 Problem Instances

The base of all problem instances is a set of well-known real-world communication networks. All of the networks use a fixed weighting for the edges acting as priority, which is either composed of attributes given by the model or picked randomly. All priority values are normalized within the interval [1, 10]. Details and properties about the problems instances are shown in Table 2.

In this table, $|V|$ and $|E|$ denote the number of vertices and edges, D the density of the graph, and K the clustering coefficient. The minimum and maximum degrees of the nodes are shown in columns $deg_{\min}$, $deg_{\max}$. All networks are undirected, such that the input degree is equal to the output degree and thus simplified to deg. The final column denotes the degree assortativity coefficient (A_{deg}). It is the measure of how likely nodes with similar degrees are connected over an edge. This is particularly useful to determine the type of network, e.g., a densely interconnected backbone network or a sparse scale-free network or spanning tree.

Table 1: LS EA parameters for optimization.

Parameter	Value
Population size	10
Number of evaluations	100 000
Experiment repetitions	100
Number of children	10
Crossover operation	Uniform crossover
Mutation operation	Bit-flip mutation
Mutation probability	0.05
Selection operation	Tournament selection
Tournament size	5
Survivor selection size	10
Local search k-value	50
Model change	after each evaluation
Model change level	5%, 10%, 25%

The first problem instance is the "National Research and Education Network (NREN) Europe" [13], aggregating information about the backbone network of the European research network in the GÉANT (formerly DANTE and TERENA) network. It is composed of the different national networks, the French RENATER, the German DFN, the Spanish RedIRIS etc. The model offers several different attributes, e.g., bandwidth, geo-location, provider, etc., which are used to form the fixed priority weighting for the optimization.

The second model, `internet-as`, is a topology graph from several autonomous systems in the internet [16]. In this model, there exist a few nodes having a high degree, while others are disconnected from the rest of the graph, i.e., having $deg = 0$. However, the graph has a comparably low density.

The last three models, `p2p-Gnutella04`, `p2p-Gnutella24`, and `p2p-Gnutella31`, are all taken from the same collection [26]. All of them form individual instances of the well-known peer-2-peer file sharing network "gnutella", which has been one of the largest active file sharing network in the past decades. These instances are especially useful as they form an overlay network over the Internet, being a logical network layer above the actual network. The gnutella networks share a common attribute of having a comparably high degree assortativity index close to 0 as opposed to the other instances. This attribute can be observed to be typical for internet service provider networks or malicious botnets.

For simulating the changes in the network models, the individual models are generated randomly on the fly. However, in order to obtain reproducibility as part of good scientific practice, the models are generated using an independent (pseudo) random number generator having the same initial seed value for all experiment repetitions. This guarantees that all consecutive models throughout all repetitions implement the same changes as the random number sequence remains the same. The random number generator used in the optimization is not affected.

Table 2: Problem instances and their attributes.

Instance	$\|V\|$	$\|E\|$	D	K	$deg_{\max}$	$deg_{\min}$	A_{deg}
NREN	1 157	1 465	0.0022	0.0610	57	1	-0.1483
internet-as	40 165	85 123	0.0001	0.0066	3 370	0	-0.1573
p2p-Gnutella04	10 879	39 994	0.0003	0.0054	103	0	-0.0083
p2p-Gnutella24	26 518	65 369	0.0001	0.0041	355	1	-0.0056
p2p-Gnutella31	62 586	147 892	0.0000	0.0039	95	1	-0.0063

6 RESULTS AND DISCUSSION

In this section, the result evaluation will be explained, followed by the actual result representation and discussion.

6.1 Result Evaluation

In general, there are two different types of result evaluation for EDO: optimality-based and behavior-based measures [24]. While optimality-based measures are used to determine the ability of the algorithm to find results having a high quality, behavior-based measures try to analyze the behavior of the algorithm with respect to different properties, e.g., diversity or robustness. In the literature, optimality-based measures are more common in EDO, as these measures focus on solving the underlying problem.

Here, we also follow an optimality-based approach called *normalized scoring* as proposed in [24]. For normalizing the fitness values over all different models and problem instances, the fitness value is divided into both initial parts: number of monitors and number of covered edges. The values are normalized within an interval of $[0, 1]$, such that the following ratios apply:

$$used\ monitors_t = \frac{monitors_t(\bar{x})}{|V_t|} \tag{6}$$

$$covered\ edges_t = 1 - \left(\frac{\sum_{\{v_i, v_j\} \in S_t} w(v_i, v_j)}{\sum_{\{v_i, v_j\} \in E_t} w(v_i, v_j)} \right) \tag{7}$$

where $S_t = \{\{v_i, v_j\} \in E_t \mid x_i = 0 \wedge x_j = 0\}$ of network model graph $G = (V_t, E_t, w)$ and candidate solution $\bar{x} \in \{0, 1\}^n$ with $|V_t|$ and $|E_t|$ being the amount of active vertices and edges at time point t. Generally, the goal of the optimization is to maximize the covered edges while minimizing the used monitors. However, displaying the individual parts of the fitness function is only used for evaluation of results. The optimization remains single-objective using the scalarized fitness value for the optimization process.

In Figure 1, for each of both measures two different results are given: LS EA and BMS LS. While LS EA measures the actual achieved optimal values of the LS EA method. The measure for BMS LS shows results of the BMS local search heuristic.

The measure Generation is a hypothetical measure indicating the potential of the remaining unevaluated individuals in the LS EA population, i.e., the individuals whose fitness value is not reevaluated. For this measure, these individuals are evaluated against the current network model, while the fitness value is neither stored nor used for further computations in the evolutionary search heuristic. As this value is close to the mean and standard deviation of the LS EA, the visualization is omitted for reasons of clarity. Numerical experimental results are provided in Table 4 for the LS EA, Table 5 for the Generation, and Table 6 for the BMS LS. The plots in the same column share the same x-axis and change level, where the change level is shown on the first and the x-axis values on the last plot in each column. Furthermore, the plots in the same row share the same y-axis and problem instance, shown on the respective first plot of each row.

In order to produce comparable results, mean and standard deviation (STD) for both measures are used over all evaluations and averaged over all experiment repetitions of the same experiment. Using this method ensures comparability of experiments using the same problem instance but different change levels. Evaluation is done individually for the LS EA, Generation, and BMS LS.

In another experiment, the aforementioned heuristics have been compared to choosing random individuals having arbitrary 50% active monitors. As the random results show equal values on all experiments having a mean of 0.7 covered edges and a standard deviation < 0.01, further numerical and visual results are omitted.

In addition to the optimality-based evaluation, a wall-clock runtime evaluation is done whose results are shown in Table 3. It provides information about the mean runtime and standard deviation (STD), both in seconds, for the optimization. This information is useful for measuring the time required by the optimization between two model changes depending on the size of the input network. However, it is strongly related to the hardware and implementation employed (which is described in Section 5.1) and only useful for relative comparison between different problem instances and experiment configurations.

6.2 Discussion

As the wall-clock runtime analysis shows, creation and evaluation of a candidate solution required < 0.1 seconds on average for all experiments so this method is generally time-wise applicable for implementation in time-critical network optimization. Obviously, the mean runtime depends highly on the size of the communication network, while implementation of the optimization has a significant influence on the differences between the change levels due to the required generation of subsequent network models.

As expected, optimization results become worse while increasing the change level on all problem instances. While the range of the average amount of monitors used is constantly between 0.50 and 0.59 on all experiments, the amount of covered edges ranges from 0.55 to 0.86 at different change levels. The effect increases the smaller the problem instance is, e.g., on problem instance NREN, the difference between the change level 5% and 25% is -0.12 (from 0.86 to 0.74) while the difference on problem instance p2p-Gnutella31

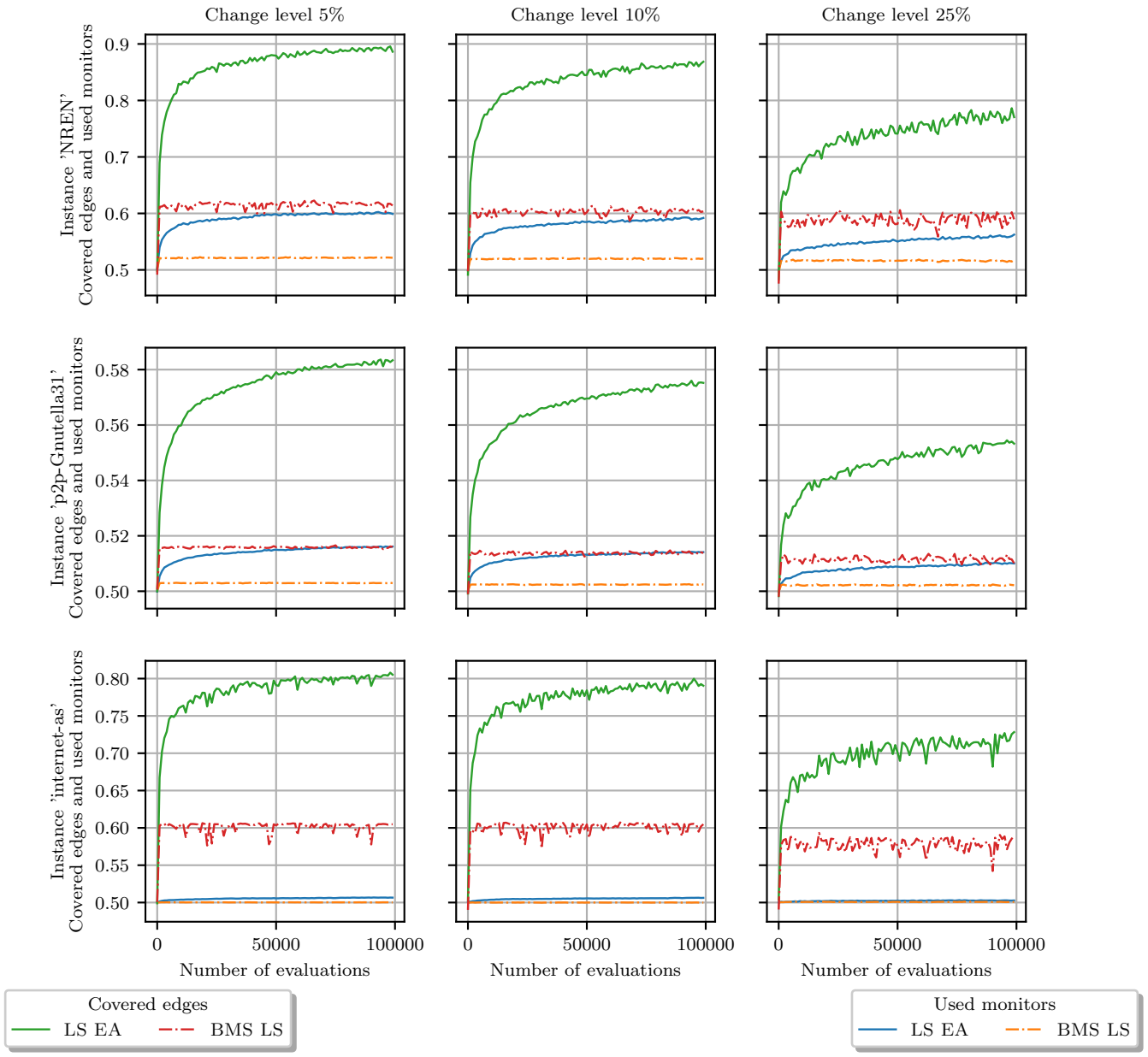

Figure 1: Runtime behavior plots for EA runs on different problem instances.

decreases by 0.02 (from 0.57 to 0.55). On all experiments, the used monitors measure starts close at a value close to 50%, as initialization is done randomly using a uniform distribution.

Comparing the results of the LS EA and the (hypothetically) missed results of the generation, it shows that both measures share the same mean but differ in standard deviation. The lower standard deviation is caused due to the higher amount of individuals used to create the measure. While for the LS EA only one value is used per experiment repetition and number of evaluation, the generation result is composed of all individuals in the population number of evaluation and repetition. However, it is surprising that obsolete individuals in the population still have a high quality.

Regarding the individual results of the local search optimization in the plots, it can be observed that for the depicted experiments no significant improvement occurs during optimization. Network model changes occur faster than the change introduced by local search using the given neighborhood relation, which might be the reason that the used local search approach does not perform well in this highly dynamic optimization context.

Table 3: Experimental results of the wall-clock runtime analysis for the optimization.

Instance	Change level	Runtime (sec.) Mean	STD
NREN	0.05	15.99	2.54
NREN	0.10	15.28	2.09
NREN	0.25	14.08	1.88
internet-as	0.05	2219.80	1408.81
internet-as	0.10	2306.65	1110.69
internet-as	0.25	2958.39	1763.95
p2p-Gnutella04	0.05	384.89	48.06
p2p-Gnutella04	0.10	455.74	163.67
p2p-Gnutella04	0.25	528.54	137.87
p2p-Gnutella24	0.05	1171.89	760.84
p2p-Gnutella24	0.10	1347.92	369.62
p2p-Gnutella24	0.25	1842.85	1312.09
p2p-Gnutella31	0.05	3934.88	2301.04
p2p-Gnutella31	0.10	4928.97	2987.07
p2p-Gnutella31	0.25	5129.20	2911.07

Table 4: Experimental results of the optimization analysis for the LS EA results.

Instance	Change level	Used Monitors Mean	STD	Covered Edges Mean	STD
NREN	0.05	0.59	0.0137	0.86	0.0233
NREN	0.10	0.58	0.0137	0.83	0.0262
NREN	0.25	0.55	0.0177	0.74	0.0421
internet-as	0.05	0.51	0.0024	0.79	0.0159
internet-as	0.10	0.51	0.0025	0.77	0.0175
internet-as	0.25	0.50	0.0029	0.70	0.0296
p2p-Gnutella04	0.05	0.54	0.0046	0.65	0.0098
p2p-Gnutella04	0.10	0.54	0.0052	0.64	0.0109
p2p-Gnutella04	0.25	0.53	0.0056	0.60	0.0141
p2p-Gnutella24	0.05	0.52	0.0031	0.61	0.0073
p2p-Gnutella24	0.10	0.52	0.0031	0.60	0.0083
p2p-Gnutella24	0.25	0.51	0.0034	0.57	0.0107
p2p-Gnutella31	0.05	0.51	0.0019	0.57	0.0053
p2p-Gnutella31	0.10	0.51	0.0021	0.57	0.0057
p2p-Gnutella31	0.25	0.51	0.0022	0.55	0.0075

Table 5: Experimental results of the optimization analysis for the Generation results.

Instance	Change level	Used Monitors Mean	STD	Covered Edges Mean	STD
NREN	0.05	0.59	0.0076	0.86	0.0135
NREN	0.10	0.58	0.0080	0.83	0.0158
NREN	0.25	0.55	0.0092	0.74	0.0240
internet-as	0.05	0.51	0.0014	0.79	0.0086
internet-as	0.10	0.51	0.0014	0.77	0.0098
internet-as	0.25	0.50	0.0016	0.70	0.0177
p2p-Gnutella04	0.05	0.54	0.0025	0.65	0.0055
p2p-Gnutella04	0.10	0.54	0.0026	0.64	0.0060
p2p-Gnutella04	0.25	0.53	0.0030	0.60	0.0077
p2p-Gnutella24	0.05	0.52	0.0016	0.61	0.0042
p2p-Gnutella24	0.10	0.52	0.0017	0.60	0.0046
p2p-Gnutella24	0.25	0.51	0.0019	0.57	0.0058
p2p-Gnutella31	0.05	0.51	0.0011	0.57	0.0029
p2p-Gnutella31	0.10	0.51	0.0011	0.57	0.0032
p2p-Gnutella31	0.25	0.51	0.0012	0.55	0.0040

Table 6: Experimental results of the optimization analysis for the BMS LS results.

Instance	Change level	Used Monitors Mean	STD	Covered Edges Mean	STD
NREN	0.05	0.52	0.0119	0.62	0.0272
NREN	0.10	0.52	0.0131	0.60	0.0332
NREN	0.25	0.52	0.0155	0.59	0.0442
internet-as	0.05	0.50	0.0027	0.60	0.0234
internet-as	0.10	0.50	0.0027	0.60	0.0278
internet-as	0.25	0.50	0.0029	0.58	0.0393
p2p-Gnutella04	0.05	0.51	0.0041	0.53	0.0073
p2p-Gnutella04	0.10	0.51	0.0045	0.53	0.0084
p2p-Gnutella04	0.25	0.51	0.0054	0.53	0.0134
p2p-Gnutella24	0.05	0.50	0.0027	0.52	0.0057
p2p-Gnutella24	0.10	0.50	0.0030	0.52	0.0075
p2p-Gnutella24	0.25	0.50	0.0035	0.52	0.0101
p2p-Gnutella31	0.05	0.50	0.0017	0.52	0.0037
p2p-Gnutella31	0.10	0.50	0.0020	0.51	0.0041
p2p-Gnutella31	0.25	0.50	0.0022	0.51	0.0060

Even though the focus of this research is not to compare the performance of different optimization methods, adequate statistical tests have been applied in order to be able to make significant statements about the performance of the local search and the hybrid LS EA. Here, the aggregated fitness value is used for comparison, as it reflects the overall performance of the respective method. As the results are non-normally distributed, the non-parametric Mann-Whitney-U test, sometimes referred to as Wilcoxon rank-sum test [17], is applied. The results of the statistical test substantiate the obvious numerical and visual results: on all experiment configurations and problem instances, the LS EA outperforms the local search significantly. As all experiment result values are within $p < 0.005$, showing that the LS EA has a higher performance than the local search, details and tables are omitted here for reasons of clarity.

7 CONCLUSION AND OUTLOOK

In this paper, the applicability of an established hybrid local search evolutionary algorithm for solving the dynamic monitor selection problem in multiple highly dynamic real-world communication infrastructure models has been studied. As described in more detail in preceding sections, a new level of automated network optimization is facilitated due to recent developments in (network) virtualization techniques. As opposed to real hardware networks, virtual software networks can programmatically accessed and managed, offering a

wide variety of, possibly automated, reactions to dynamic changes. A major aspect of this research has been addressing the volatility of the network instances reflecting changes in real networks. The focus of this paper has been to show the general feasibility of the presented approach for dynamic network optimization on several different real-world network problem instances.

As the results show, the used hybrid evolutionary search heuristics are applicable in even difficult circumstances, where significant changes of the underlying network occur. Results are stagnating with an increasing amount of change introduced in the problem instances. Especially compared to the current situation, requiring a manual interaction of network administration staff, the proposed approach offers great benefits with respect to reaction rate, threat prevention, and risk mitigation.

This research forms the base for multiple interesting future aspects, as automated network design and optimization will become an indispensable part of future network management. As the underlying problem is time-critical, tuning the methods and method parameters in order to improve optimization performance is crucial and will be intensified in the future, while extending the number of problem instances will help to generalize the proposed method. Application of variable-length encoding could be one interesting aspect here, which additionally offers independence of the optimization from a base model. Furthermore, an extensive comparison of further methods, e.g., using aforementioned NuMVC, FastVC or a (1+1) EA, could be interesting in order to find the most suitable heuristic for optimization of the given problem.

ACKNOWLEDGMENTS

This work was supported by the Spanish Ministry of Economy and Competitiveness (National Program for Research, Development and Innovation) and with funds of the European Union (European Regional Development Fund - ERDF), project DArDOS TIN2015-65845-C3-3-R and Excellence Network SEBASENet TIN2015-71841-REDT. Responsible for the content are the authors.

REFERENCES

[1] Jürgen Branke and Hartmut Schmeck. 2003. *Designing Evolutionary Algorithms for Dynamic Optimization Problems.* Springer Berlin Heidelberg, Berlin, Heidelberg, 239–262. https://doi.org/10.1007/978-3-642-18965-4_9

[2] Shaowei Cai. 2015. Balance between complexity and quality: Local search for minimum vertex cover in massive graphs. *IJCAI International Joint Conference on Artificial Intelligence* 2015-Janua, Ijcai (2015), 747–753.

[3] Shaowei Cai, Jinkun Lin, and Chuan Luo. 2017. Finding A Small Vertex Cover in Massive Sparse Graphs: Construct, Local Search, and Preprocess. *Journal of Artificial Intelligence Research* 59 (2017), 463–494.

[4] Ankit Chauhan, Tobias Friedrich, and Francesco Quinzan. 2017. Approximating Optimization Problems using EAs on Scale-Free Networks. In *Genetic and Evolutionary Computation Conference (GECCO)*. 235–242. https://doi.org/10.1145/3071178.3071257 arXiv:1704.03664

[5] Jianer Chen, Iyad A. Kanj, and Ge Xia. 2010. Improved upper bounds for vertex cover. *Theoretical Computer Science* 411, 40-42 (sep 2010), 3736–3756. https://doi.org/10.1016/J.TCS.2010.06.026

[6] Carlos Artemio Coello Coello. 2012. Constraint-handling techniques used with evolutionary algorithms. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion - GECCO Companion '12 (GECCO '12)*. ACM, New York, NY, USA, 849. https://doi.org/10.1145/2330784.2330920

[7] C. Cruz, J.R. González, and D.A. Pelta. 2011. Optimization in dynamic environments: A survey on problems, methods and measures. *Soft Computing* 15, 7 (2011), 1427–1448. https://doi.org/10.1007/s00500-010-0681-0

[8] Mou Dasgupta, G. P. Biswas, and Chandan Bhar. 2012. Optimization of multiple objectives and topological design of data networks using genetic algorithm. In *2012 1st International Conference on Recent Advances in Information Technology (RAIT)*. IEEE, 256–262. https://doi.org/10.1109/RAIT.2012.6194516

[9] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. 2012. Multiplicative drift analysis. *Algorithmica* 64, 4 (2012), 673–697. https://doi.org/10.1007/s00453-012-9622-x arXiv:1101.0776

[10] Mitsuo Gen and Runwei Cheng. 1999. *Genetic Algorithms and Engineering Optimization*. Vol. 7. John Wiley & Sons. 512 pages. https://doi.org/10.1002/9780470172261

[11] ITU. 2014. *The World in 2014 - ICT Facts and Figures.* Technical Report in 2008. International Telecommunication Union (ITU), Geneva, Switzerland. 8 pages. https://doi.org/10.1787/9789264202085-5-en

[12] Richard M. Karp. 2010. Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. 219–241. https://doi.org/10.1007/978-3-540-68279-0_8 arXiv:arXiv:1011.1669v3

[13] Simon Knight, Nickolas Falkner, Hung X. Nguyen, Paul Tune, and Matthew Roughan. 2012. I can see for miles: Re-visualizing the internet. *IEEE Network* 26, 6 (2012), 26–32. https://doi.org/10.1109/MNET.2012.6375890

[14] Sofiane Lagraa and JéRome François. 2017. Knowledge discovery of port scans from darknet. In *Proceedings of the IM 2017 - 2017 IFIP/IEEE International Symposium on Integrated Network and Service Management*. IEEE, 935–940. https://doi.org/10.23919/INM.2017.7987415

[15] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. 2008. Kronecker Graphs: An Approach to Modeling Networks. *The Journal of Machine Learning Research* 11 (2008), 985–1042. https://doi.org/10.1145/1756006.1756039 arXiv:0812.4905

[16] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2006. Graph Evolution: Densification and Shrinking Diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (mar 2006). https://doi.org/10.1145/1217299.1217301 arXiv:physics/0603229

[17] H. B. Mann and D. R. Whitney. 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics* 18, 1 (1947), 50–60. https://doi.org/10.1214/aoms/1177730491

[18] Antonio Marotta, Fabio D'Andreagiovanni, Andreas Kassler, and Enrica Zola. 2017. On the energy cost of robustness for green virtual network function placement in 5G virtualized infrastructures. *Computer Networks* 125 (oct 2017), 64–75. https://doi.org/10.1016/J.COMNET.2017.04.045

[19] Zbigniew Michalewicz, Martin Schmidt, Matthew Michalewicz, and Constantin Chiriac. 2007. Adaptive business intelligence: Three case studies. In *Studies in Computational Intelligence*. Vol. 51. Springer, Berlin, Heidelberg, 179–196. https://doi.org/10.1007/978-3-540-49774-5_8

[20] Robin Mueller-Bady, Martin Kappes, Lukas Atkinson, and Inmaculada Medina-Bulo. 2017. Multijob. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '17*. ACM, Berlin, 1231–1238. https://doi.org/10.1145/3067695.3082476

[21] Robin Mueller-Bady, Martin Kappes, Inmaculada Medina-Bulo, and Francisco Palomo-Lozano. 2017. Optimization of Monitoring in Dynamic Communication Networks using a Hybrid Evolutionary Algorithm. In *GECCO'17: Proceedings of the 2017 conference on genetic and evolutionary computation companion*. https://doi.org/10.1145/3071178.3071255

[22] P.S.R. Murty and P.S.R. Murty. 2017. *Graph Theory* (5th ed.). Springer, Heidelberg. 7–17 pages. https://doi.org/10.1016/B978-0-08-101111-9.00002-1 arXiv:arXiv:1011.1669v3

[23] Ferrante Neri and Raino A.E. Mäkinen. 2007. Hierarchical evolutionary algorithms and noise compensation via adaptation. In *Studies in Computational Intelligence*. Vol. 51. Springer, 345–369. https://doi.org/10.1007/978-3-540-49774-5_15

[24] Trung Thanh Nguyen, Shengxiang Yang, and Juergen Branke. 2012. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation* 6 (oct 2012), 1–24. https://doi.org/10.1016/j.swevo.2012.05.001

[25] F.P. Quintao, F.G. Nakamura, and G.R. Mateus. 2005. Evolutionary Algorithm for the Dynamic Coverage Problem Applied to Wireless Sensor Networks Design. In *2005 IEEE Congress on Evolutionary Computation*, Vol. 2. IEEE, 1589–1596. https://doi.org/10.1109/CEC.2005.1554879

[26] Matei Ripeanu and Ian Foster. 2002. Mapping the Gnutella Network: Macroscopic Properties of Large-Scale Peer-to-Peer Systems. *IEEE Internet Computing Journal* (2002), 85–93. https://doi.org/10.1007/3-540-45748-8_8 arXiv:cs/0209028

[27] Philipp Rohlfshagen and Xin Yao. 2008. Attributes of Dynamic Combinatorial Dynamic Optimisation : A Brief Overview. In *Simulated Evolution and Learning*, Xiaodong Li, Michael Kirley, Mengjie Zhang, David Green, Vic Ciesielski, Hussein Abbass, Zbigniew Michalewicz, Tim Hendtlass, Kalyanmoy Deb, Kay Chen Tan, Jürgen Branke, and Yuhui Shi (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 442–451.

[28] Ole Tange. 2011. GNU Parallel: the command-line power tool. *;login: The USENIX Magazine* 36, 1 (feb 2011), 42–47. https://doi.org/10.5281/zenodo.16303

[29] Shengxiang Yang, Hui Cheng, and Fang Wang. 2010. Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews* 40, 1 (jan 2010), 52–63. https://doi.org/10.1109/TSMCC.2009.2023676