Journal of Universal Computer Science, vol. 25, no. 7 (2019), 788-815
submitted: 18/10/18, accepted: 29/4/19, appeared: 28/7/19 © J.UCS

On the Automated and Reactive Optimization of Highly-Dynamic Communication Network Infrastructures

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

Inmaculada Medina-Bulo, Francisco Palomo-Lozano
(University of Cádiz, Spain
[inmaculada.medina,francisco.palomo]@uca.es)

Abstract: In this paper, the applicability of heuristic methods for an automated and reactive optimization of network infrastructures in highly-dynamic communication networks is studied. With an increasing amount of (mobile) participants and at the same time significantly growing quality requirements in communication networks over the past years, optimization of communication infrastructures will become an inevitable challenge in providing a reliable and high-quality communication service. Mostly, changes in highly-dynamic networks, which may be planned or unplanned, happen swiftly, such that it is not possible to apply manual optimization. Thus, an automated and reactive optimization becomes necessary to address this problem. Two major issues arise from the optimization of highly-dynamic communication networks. First, the complexity of problems, which is either implied by the complex optimization problem or the number of different possibly concurrent goals subject to optimization. Second, the highly-dynamic optimization search space, where network topologies may change rapidly introducing severe challenges for the optimization process.

Here, different evolutionary and greedy optimization heuristics for the optimal selection of monitors in communication networks are studied and compared. Monitor selection is a well-known, important, and complex ($\mathcal{NP}$-hard) optimization problem, serving as a current and actual use case for the general concept of highly-dynamic communication network optimization. As the results show, two of three methods reliably provide solutions of sufficiently high quality in reasonable time, enabling the applicability of heuristic methods of optimization in highly-dynamic communication networks. Results of the experiments are obtained using state-of-the-art statistical methods for evaluation of (evolutionary) search heuristics on a set of 39 real-world and synthetic benchmark problem instances.

Key Words: Evolutionary computation, greedy algorithms, computer networks, communication system security
Category: C.2.1 C.2.3

1 Introduction

Over the past decades, communication infrastructures have grown massively both in quantity and quality. Today, network service providers are challenged by an increasing number of communication devices in the area of, e.g., mobile- or internet-of-things communication, and a rising requirement for connections of high quality, e.g., reliable and fast data transfer with low latencies. Especially in the area of mobile communication using smartphones, tablets, laptops, or other devices connected via mobile-broadband subscriptions, the global number of participants increased over more than 20% annually in the past 5 years and it is expected to reach a globally total of 4.3 billion according to the statistics of the International Telecommunication Union [ITU, 2013].

In the area of network management it is necessary to distinguish between two different types of events which may occur during operation: ordinary (planned) events and extraordinary (unplanned) events. In most cases, ordinary events such as roaming of mobile clients or maintenance of network infrastructure devices are taken into account during the design of the network. Requirements may change over time, e.g., due to a higher number of mobile roaming devices or new (quality) obligations, but may usually be covered by flexible and scalable design beforehand. On the other side, extraordinary events such as hardware failures or attacks on the infrastructure may as well be addressed by proper prevention techniques, e.g., redundancy and countermeasures for security incidents. However, due to the financial effort, rapidly evolving area of possible attacks and vulnerabilities, and an increasing complexity of the systems, ultimate security is neither achievable nor efficient. Hence, in order to react reasonably and efficiently on changes in the network infrastructure, an automated, reactive and steady optimization of the network infrastructure offers huge advantages.

Optimizing communication networks is subject to research since years. However, with the evolving use and importance of networks as described before and the developing technology of *Software Defined Networking (SDN)* and *Network Function Virtualization (NFV)*, a completely new way of network management becomes possible [Marotta et al., 2017]. While the focus of network optimization had been on creating robust, resilient and efficient design and maintenance of stationary network topologies, SDN and NFV enable a bird's-eye view on the whole administrative domain and additionally offer programmatic access to network properties such as bandwidth management or dynamic routing. This also facilitates new network security applications by enabling early detection of possible threats, e.g., botnets, denial of service attacks, or propagation of malware [Lagraa and Francois, 2017].

In this paper, the problem of monitor placement in highly-dynamic communication infrastructures is addressed using different methods from the area of *Evolutionary Algorithms (EAs)* and other heuristic optimization techniques. As monitors act as sensors for traffic in communication networks, it is the most basic but also possibly most important step towards securing communication network infrastructures. However, while being an important problem in network management, the monitor selection is just one possible use case of reactive network optimization, which might be subject to change depending on the corresponding requirements.

One of the major challenges addressed here is the highly-dynamic infrastructure, which implies steady and frequent changes of the network topology, such that only a limited time-window is available for the optimization process. In this case, the network may change after each evaluation of the optimization method, which has several implications for the optimization process which are described in more detail in the following sections. However, one has to distinguish between dynamic monitoring and dynamic infrastructures. While dynamic monitoring describes the dynamic placement of monitors in a network, dynamic infrastructures describe the changing topology of a network to be monitored. It is important to mention that it can be useful to apply dynamic monitoring even in static infrastructures as network traffic is usually dynamic and hence requirements for monitoring change accordingly.

This paper introduces the following major contributions in the area of automated and reactive optimization of highly-dynamic communication network infrastructures:

- Study of the general applicability of heuristic methods in reactive optimization of highly-dynamic communication networks
- Comparison of three different current methods for optimization of monitoring in dynamic communication networks
- Consideration of 37 well-known real-world and synthetic problem instances for benchmarking the optimization methods
- Conduction of statistical significant experiments in order to show advantages and disadvantages of the methods under study
- Runtime analysis of the optimization methods in order to show applicability

The remainder of this paper is structured as follows: In the following section, an overview of the current state of the art is given, followed by the definition of the monitor selection problem in Section 3 and the proposed solving strategy in Section 4. Section 5 describes the experimental setup for the experiments conducted, while Section 6 discusses the results. The paper closes drawing conclusions from the study and providing an outlook for possible future work in Section 7.

2 Related Work

Optimization of communication networks has been done in the past, especially in relation to the optimization of mathematical graphs [Yang et al., 2008, Chiang, 2009, Baccelli and Zuyev, 1999, Atiqullah and Rao, 1993]. Most of the research focuses on the network design problem and its variants, i.e., the initial design of a network as opposed to a steady optimization. Recently, the development of network virtualization techniques like SDN and NFV enables a completely different way to manage networks providing programmatic access to detailed network and traffic configurations and thus facilitates automated and reactive network optimization. This feature of SDN and NFV can be used in conjunction with heuristic optimization in order to implement a continuous optimization lifecycle, i.e., SDN provides current information for the optimization method, then the optimizer provides an improved model of the network which is finally rolled out into the actual network using SDN again.

While changes in communication infrastructures may have several reasons, e.g., hardware failure, malfunction, non-stationary participant, etc., SDN and NFV add another challenge for optimization of such dynamic infrastructures. Hence, supporting processes of networks, e.g., monitoring, need to be adapted in order to keep pace with a changing requirements in dynamic communication networks.

In the area of EAs, *Evolutionary Dynamic Optimization (EDO)* [Branke and Schmeck, 2003, Rohlfshagen and Yao, 2008] is an important field covering dynamic search spaces in optimization. It has grown over the past decades, as shown by the survey papers of Nguyen et al. [Nguyen et al., 2012] and Cruz et al. [Cruz et al., 2011]. EDO has been applied to different areas, such as car distribution systems [Michalewicz et al., 2007] or electrical grounding grids [Neri and Maekinen, 2007]. Furthermore, EDO has also been applied to network-based problems, such as wireless sensor network design [Quintao et al., 2005], dynamic shortest-path routing problems in mobile ad-hoc networks [Shengxiang Yang et al., 2010] or in stationary network environments such as the topological design problems having multiple objectives [Dasgupta et al., 2012].

Current research work of different researchers disagree on which method is the most beneficial for solving graph-based problems, particularly the monitor selection problem. The most efficient implementations for solving the monitor selection problem, which is strongly related to the well-known *minimum vertex cover (MVC)* problem, are NuMVC [Cai et al., 2017] and FastVC [Cai, 2015] as proposed by Cai, which are both based on a greedy local search and studied on experiments. However, Chauhan et al. have shown that an evolutionary approach is generally more efficient than a local search [Chauhan et al., 2017] for solving the MVC problem in stationary scale-free networks.

All approaches discussed are applied to static graph problem instances, e.g., the exact optimization method proposed by Chen [Chen et al., 2010], while the problem studied in this paper covers optimization in highly-dynamic networks. Hence, using a hybrid method combining both, EAs and local search, seems to be a reasonable choice.

Solving the monitor selection problem in highly-dynamic network environments has first been addressed by Mueller-Bady et al. in [Mueller-Bady et al., 2018]. Based on this idea, this paper introduces some major improvements for solving the monitor selection problem in highly-dynamic networks:

- Experiments are extended. Here, another optimization method, a common EA, is used in order to complete the comparison of the hybrid local search EA, the standalone local search method, and the common EA.
- The number and type of problem instances is significantly increased. A set of 39 diverse problem instances are taken from both, real-world networks and well-known synthetic graph benchmarks in order to be able to evaluate a general applicability of the proposed methods to highly-dynamic problem instances.
- A state-of-the-art parameter tuning method is applied in advance to tweak the parameters of the optimization method, taking into account the different problem instances and optimization parameters, and
- A comprehensive and significant statistical analysis is applied to the results of the experiments and displayed in various formats.

3 The Monitor Selection Problem

The first step in securing a communication network is implementing an adequate monitoring in order to be able to evaluate events in the network and, in case it is necessary, implement suitable countermeasures. In network management and surveillance, monitoring is usually accomplished by installing monitors on specific nodes in the target network, such that traffic that passes through edges connected to these nodes, often referred to as links in terms of communication networks, can be analyzed. The whole network is said to be monitored in case all active links have at least one active adjacent monitor capturing traffic.

In general, both types of network infrastructure changes might appear during ongoing monitoring: software changes induced by, e.g., SDN, or hardware changes due to failure or malfunction. For monitoring a network, both changes can make it necessary to adapt positions of monitoring nodes to the same extent. While malfunction of hardware occurs unexpected even during an active connection but usually with a low frequency, changes induced by SDN may have a very high frequency depending on the network. Hence, for the remainder of this paper, the actual reason for monitoring adaptions in the network is not further considered in more detail. However, in order to stress the optimization methods, a worst case with a high frequency of changes is assumed for the experiments.

A fast and comprehensive method to monitor all links in a network is to implement monitors on all nodes in the network. However, as deployment of a monitor also implies several drawbacks such as implementation and maintenance cost or performance loss on network nodes, it is desirable to implement the optimal number of monitors at the optimal positions in the network in order to cover all links. In the problem under study, each node in the network may be used as a monitor. However, in case the particular use case makes it necessary, further constraints on this model may restrict the use of nodes as monitors based on the, e.g., type, occupation, or cost for implementation.

In order to model communication network infrastructures, mathematical graphs have been shown to be a beneficial. In this paper it is assumed that a communication network is modeled by a finite simple graph, such that $G = (V, E)$, 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. Throughout this paper, unless otherwise mentioned, definitions of Diestel apply [Diestel, 2017].

As described before, the monitor selection problem is closely related to the well-known $\mathcal{NP}$-hard MVC optimization problem introduced by Karp [Karp, 1972]. A vertex cover 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. A minimum vertex cover 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 the sake of brevity, the proof is omitted here.

As opposed to the MVC problem, the dynamic monitor selection problem uses an additional edge weighting in order to prioritize monitoring on specific links in the network. With the edge priority it is possible to model monitoring priorities depending on different measures, e.g., graph centrality. In the experiments conducted, edge priorities are either derived from information available from the problem instances or they are generated randomly. In order to apply the edge priority to the network model, the definition is extended to $G = (V, E, w)$ with $w : E \to \mathbb{N}$ being a non-negative weighting function modeling the priority.

In order to reflect the time constraints in the dynamic monitor selection problem, the optimization is done within the period $[t_{start}, t_{end}]$ with t_{start} being the start and t_{end} being the end of the optimization process. Thus, the quality of a solution depends on which edges are covered, the amount of monitors used and the time point at which the solution was acquired. With the definition of the time window, the previously defined network model graph reflecting the problem instance can be defined as dynamic graph $G_t = (V_t, E_t, w)$ with $|V_t|$ and $|E_t|$ being the amount of active vertices and edges at time point t and the time-independent priority function w. As the underlying search space is highly-dynamic, the resolution of the time windows is the number of evaluations of the optimization method, i.e., the network model changes after each evaluation of the target network.

4 Proposed Solving Strategy

In this section, the methods for solving the monitor selection problem are described in necessary detail. Generally, the proposed solving strategy is a hybrid search heuristic composed of a robust EA for broad exploration of the search space and a fast local search approach for exploitation of the (local) search space, forming the *local search evolutionary algorithm (LS EA)*. The method has been successfully applied in a different context before [Mueller-Bady et al., 2017b]. However, in this paper, the proposed LS EA is compared to its individual components, i.e., the LS and the EA part. In the following, these components are described in detail.

4.1 Evolutionary Algorithm

The evolutionary component is based on a generational EA. As common for this kind of heuristics, a set of random initial candidate solutions (individuals) is drawn and evaluated, forming the initial population. Then, until a certain termination condition is satisfied, the evolutionary process is applied iteratively as follows. First, two distinct individuals are selected from the population acting as parents. These parents create two children using a recombination operation which are mutated introducing minor random changes to the candidate solutions. Finally, the new population is are reduced to its original size using selective pressure induced by a selection operation. The termination condition for the optimization process is usually that a certain number of evaluations or a specified quality threshold has been reached.

According to Gen et al. [Gen and Cheng, 1999] and Doerr et al. [Doerr et al., 2012], there exist different formats for representing encodings of candidate solutions. For the dynamic monitor selection problem, a vertex-based approach is used 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:

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

In order to determine the quality of such an individual, first the number of monitors is determined:

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

where a lower amount of monitors indicates a higher quality.

However, as individuals may also represent infeasible solutions, i.e., there exist one or multiple edges in the graph not being covered, solutions not fulfilling these requirements are penalized using a linear distance function [Coello Coello, 2012] based on the given edge weighting w:

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

where $S = \{\{v_i, v_j\} \in E \mid x_i = 0 \wedge x_j = 0\}$. Finally, the result is combined 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 $\bar{x}' \in \{0,1\}^n$, such that $f(\bar{x}') \leq f(\bar{x}), \forall \bar{x} \in \{0,1\}^n$.

As evaluation of candidate solutions is done in dynamic problem instances, it is necessary to reflect time of the optimization process within optimization period $[t_{start}, t_{end}]$ in the evaluation of the fitness of the individuals:

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

In this paper, the optimization period is defined as the number of evaluations from the beginning of the optimization process, such that 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 solution and a neighborhood relation. In most cases, local search is applied iteratively on all existing neighbors until no better solution can be found while replacing the currently searched candidate solution with the possibly improved one. In this specific case the neighborhood relation is set to a Hamming distance of 1, i.e., two neighbors differ at exactly one position in its representation [Hamming, 1950].

While for some problems an exhaustive search of the whole neighborhood might be applicable, it is not the case here. In many of the problem instances under study, it is not even possible to do a single iteration of local search as the size of the problem instance and therefore its encoding as individual is too large. Hence it is necessary to introduce a parameter for restricting the search process in the neighborhood. Here, this is done introducing a new parameter,k, which restricts the amount of searched neighbors to k, leading to a variation of the *Best-from-Multiple Selection (BMS)* as proposed by Cai [Cai, 2015]. While Cai uses the BMS heuristic for choosing a possible candidate node for removal from an actual vertex cover as part of the FastVC algorithm, BMS is used here for finding possible improved candidate solutions. The whole scheme of the BMS LS is shown in Algorithm 1.

Algorithm 1 Scheme of the best-of-multiple local search method (BMS LS)

```
Require: Initial individual ind, parameter k, comparison function f
  function LOCALSEARCH(ind, k, f)
      currentOpt ← ind
      repeat
          neighbors ← K_RANDOM_NEIGHBORS(currentOpt, k)
          bestNeighbor ← OPTIMUM(neighbors)
          if f(bestNeighbor) < f(currentOpt) then
              currentOpt ← bestNeighbor
          end if
      until bestNeighbor ≠ currentOpt
      return currentOpt
  end function
```

4.3 LS EA

In order to combine the previously introduced EA and BMS LS methods, both methods are aggregated in the hybrid LS EA. After initialization of the population, the current best individual from the population is used for the local search. As soon as no further improvement can be made using the BMS LS on this individual, the best found candidate solution by the local search is added back to the population where the usual evolutionary search process continues. For each iteration of the EA, first the local search is applied followed by the evolutionary search process until the termination condition is satisfied. For the experiments conducted, the parameters used for both, the evolutionary and the local search, are described in upcoming Section 5.

4.4 Research Questions and Hypotheses

Related to the problem as described before, the following research questions have been developed including the derived hypotheses in order to measure the performance of the proposed solving strategies.

RQ1 Is it possible to optimize highly-dynamic communication network infrastructures using a heuristic search method?

RQ2 Does the proposed EA improve the solutions compared to a common BMS LS in the optimization of highly-dynamic communication networks?

RQ3 Does the proposed LS EA improve the solutions compared to a common BMS LS in the optimization of highly-dynamic communication networks?

RQ4 Does the proposed LS EA improve the solutions compared to a common EA in the optimization of highly-dynamic communication networks?

The following hypotheses are used in order to scientifically provide answers to the research questions with the help of the experiments conducted in this section:

$\mathbf{H1_1}$ The median fitness of the best solutions is improved using an EA for optimization of highly-dynamic communication networks as opposed to using a BMS LS heuristic.

$\mathbf{H2_1}$ The median fitness of the best solutions is improved when a LS EA is used for optimization of highly-dynamic communication networks as opposed to using a BMS LS heuristic.

$\mathbf{H3_1}$ The median fitness of the best solutions is improved when a LS EA is used for optimization of highly-dynamic communication networks as opposed to using an EA heuristic.

5 Experiments

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

5.1 Experimental Setup

Parameters are tuned in accordance with the racing parameter tuning method *irace* [Lopez-Ibanez et al., 2016], while all instances are taken into account for finding the parameter configuration (training set). Results of the parameter tuning are shown in Table 1 as numerical and categorical values. Table 1 shows the particular values used for the optimization process.

The experiments conducted in this paper follow a $(\mu + \lambda)$ EA scheme [Beyer and Schwefel, 2002] using the parameters as optimized by the *irace* method. Hence, selection of survivors is done from both, the 251 existing parents and the 251 children. For selecting both, parents and survivors, tournament selection with a tournament size of 4 is used. To perform this selection, a uniformly distributed set of individuals from the population of size 4 (tournament size) is drawn, while the best out of all tournament participants is finally selected. As selection is done twice during one iteration, for selecting the parents and the survivors. The tournament size parameters help to reduce selective pressure on the population as high selective pressure may be disadvantageous for the search process by inducing premature convergence of the population to a possible local optimum.

As variation operators, uniform crossover is used as recombination method for creating new offspring of the parents. Crossover is applied uniformly over all elements of the individual with a certain percentage (here: 50%). This leads to children implementing individual parts of their parents, while specific contents and positions of elements which are equal for both parents remain the same for the children.

In order to introduce minor random changes, bit-flip mutation is used as mutation variation operator. Here, every element of each child is mutated using a mutation probability of 0.001. The mutation probability found by the parameter tuning is comparably small, leading to the observation that mutation does not seem to have a high impact on the optimization process.

According to the *irace* tuning method, the local search parameter k can not be observed to have the same impact on the result quality as other factors like population or tournament size. The optimized value for parameter k was found to be $k = 1$, such that one random neighbor of each LS iteration is searched for possible improvement.

In this experiment, the amount of different network models is fixed to the number of evaluations per optimization run (100 000) while the model volatility is a variable parameter describing the amount of change applied in every evaluation, simulating the dynamics of the network. In order to simulate different activity levels in the network, the change of the network is divided into different model volatility levels from a base model: 5%, 10%, and 25%. Having a volatility level of 5% implies that arbitrary (randomly chosen) 5% nodes from the given base model are toggled from inactive to active or vice versa, while inactive nodes 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. Inactive nodes may be switched back to active in the next iteration of the model change. Weightings of active edges are preserved over all changes between different network models.

The number of evaluations is restricted to 100 000 as termination condition for all methods under study. In order to obtain results of statistical significance, each experiment is repeated 100 times per configuration. For reporting the results, median and *interquartile range (IQR)* are used for all experiments, while mean and *standard deviation (SD)* are used for summarizing execution times.

The experiments' results are obtained using the parallelization framework "multijob" [Mueller-Bady et al., 2017a] and GNU parallel [Tange, 2011] on a cluster configuration with two servers, where the first is composed of 2x Intel Xeon

Table 1: Optimization parameters according to parameter tuning method *irace*.

Parameter		Value
Evolutionary algorithm		
Population size		251
Number of children	(λ)	251
Number of survivors	(μ)	251
Number of evaluations		100 000
Experiment repetitions		100
Selection operation		Tournament selection
Tournament size		4
Survivor selection strategy		$(\mu + \lambda)$
Crossover operation		Uniform crossover
Mutation operation		Bit-flip mutation
Mutation probability		0.001
Number of models		100 000
Model volatility		$0.05, 0.10, 0.25$
Local search		
Local search parameter k		1

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, and additional 10 desktop machines each having one Intel Core i7-6700K @ 4.00 GHz (4 cores, 8 hyperthreads) and 16GB RAM. All machines use a Debian Linux as operating system on the machine without virtualization techniques, which is important for comparing wall-clock runtime results. The implementation of the experiments is done in the C programming language and compiled using the GCC 6.4.0 compiler.

5.2 Problem Instances

The problem instances, i.e., instances of different communication network, are divided into two major parts. The first part contains benchmark instances from the well-known DIMACS, DIMACS10 and BHOSLIB libraries, which were obtained from the network data repository [Rossi and Ahmed, 2015]. Those benchmarks are interesting for testing the overall performance of the method proposed, but as those are synthetic network graph topologies, results are not applicable to real-world network scenarios without further considerations. Weights are assigned random values in the $[2, 10]$ interval for each problem instance in case no other priority is given. Hence, leaving an edge uncovered introduces an increase of the fitness value of at least 2, while covering a node with a monitor increases the fitness value by just 1. The selection pressure of the evolutionary process will therefore favor solution covering all edges instead of reducing the number of monitors (as a first step), which is the desired output for a swift optimization of monitors in exceptional situations. Link priorities are assigned once, such that they remain constant over all experiments shown in this document.

The second part of the problem instances contains models of real-world communication networks. The graph model of the `NREN` Europe [Knight et al., 2012] is a composition of all core routers of the European research network. This dataset provided, in addition to topology information, also information about geopositions, available bandwidth, etc. The latter is used to generate the priorities for existing links in this network instance as a linear mapping from the bandwidth range (minimum to maximum available bandwidth) to the set of priorities {2, ..., 10}, i.e., the lowest bandwidth has priority 2 while the highest bandwidth has priority 10.

Multiple instances of the well-known `p2p-Gnutella` Peer-2-Peer file sharing overlay network [Ripeanu and Foster, 2002, Leskovec et al., 2007] are used as problem instances. Those networks are interesting as they reflect a specific case of virtual overlay networks in communication networks. Further real-world network instances are used for benchmarking the methods proposed, e.g., the `internet-as` instance, which is a model of the different autonomous systems used in the Internet to transfer and transit data flows.

All benchmarks differ in size, density and graph attributes, e.g., degrees, and clustering coefficients. In Table 2, all details of the total 39 problem instances can be found, where $|V|$ denotes the number of vertices, $|E|$ the number of edges and D the density measure of the graph. Additionally, the clustering coefficient ($clus$) [Holland and Leinhardt, 1971], the minimum and maximum degree of the graph ($deg_{\min}, deg_{\max}$) and the degree assortativity coefficient (A_{deg}) [Newman, 2003] are shown, all of which are indicators for the practical applicability of the model to real communication networks.

6 Results and Discussion

In this section, the result evaluation will be explained, followed by the actual results in different representation formats and a discussion. For the sake of completeness, the whole dataset and an edited document including comprehensive tables is provided online.[1] Due to restrictions in the number of pages and for reasons of clarity, the tables are not provided in this paper.

[1] https://github.com/rmuellerb/DynamicOptimizationData

Table 2: Problem instances and their attributes.

INSTANCE	$\lvert V\rvert$	$\lvert E\rvert$	D	*clus*	$deg_{\max}$	$deg_{\min}$	A_{deg}
NREN	1 157	1 465	0.0022	0.0610	57	1	-0.1483
tech-routers-rf	2 114	6 632	0.0015	0.2283	109	0	-0.0529
tech-WHOIS	7 477	56 943	0.0010	0.3062	1 079	0	-0.0218
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-Gnutella05	8 846	31 839	0.0004	0.0076	88	1	-0.0054
p2p-Gnutella06	8 717	31 525	0.0004	0.0081	115	1	-0.0032
p2p-Gnutella08	6 301	20 777	0.0005	0.0207	97	1	-0.0285
p2p-Gnutella09	8 114	26 013	0.0004	0.0172	102	1	-0.0327
p2p-Gnutella24	26 518	65 369	0.0001	0.0041	355	1	-0.0056
p2p-Gnutella25	22 687	54 705	0.0001	0.0045	66	1	-0.0062
p2p-Gnutella30	36 682	88 328	0.0001	0.0052	55	1	-0.0214
p2p-Gnutella31	62 586	147 892	0.0000	0.0039	95	1	-0.0063
delaunay_n10	1 000	3 100	0.0058	0.3818	12	0	-0.2096
delaunay_n11	2 000	6 100	0.0029	0.3786	13	0	-0.2180
delaunay_n12	4 100	12 300	0.0015	0.3791	14	0	-0.2125
delaunay_n13	8 200	24 500	0.0007	0.3796	12	0	-0.2161
delaunay_n14	16 400	49 100	0.0004	0.3794	16	0	-0.2248
delaunay_n15	32 800	98 300	0.0002	0.3791	18	0	-0.2226
delaunay_n16	65 537	196 575	0.0000	0.3793	17	0	-0.2237
delaunay_n17	131 073	393 176	0.0000	0.3795	17	0	-0.2219
delaunay_n18	262 145	786 396	0.0000	0.3795	21	0	-0.2261
delaunay_n19	524 289	1 572 823	0.0000	0.3797	21	0	-0.2235
delaunay_n20	1 048 577	3 145 686	0.0000	0.3797	23	0	-0.2282
frb30-15-1	450	83 200	0.8235	0.8210	407	0	-0.4874
frb30-15-2	450	83 200	0.8230	0.8204	404	0	-0.4937
frb30-15-3	450	83 200	0.8237	0.8214	400	0	-0.4871
frb30-15-4	450	83 200	0.8235	0.8211	401	0	-0.4728
frb30-15-5	450	83 200	0.8239	0.8221	403	0	-0.4833
frb35-17-1	595	148 900	0.8424	0.8409	544	0	-0.4901
frb35-17-2	595	148 900	0.8424	0.8404	541	0	-0.4916
frb35-17-3	595	148 900	0.8419	0.8412	549	0	-0.4960
frb35-17-4	595	148 900	0.8424	0.8415	560	0	-0.4874
frb35-17-5	595	148 900	0.8407	0.8393	550	0	-0.4846
frb40-19-1	761	247 106	0.4273	0.8561	703	0	-0.4925
frb40-19-2	761	247 157	0.4273	0.8559	702	0	-0.4967
frb40-19-3	761	247 325	0.4276	0.8558	702	0	-0.4917
frb40-19-4	761	246 815	0.4267	0.8542	692	0	-0.4979
frb40-19-5	761	246 801	0.4267	0.8543	691	0	-0.4976

6.1 Result Evaluation

In order to evaluate the results of dynamic search spaces, especially but not limited to evolutionary search heuristics, it can be distinguished between optimality-based and behavior-based measures as shown by Nguyen [Nguyen et al., 2012]. The former measure determines the ability of a problem solving method to reach a specific solution quality, while the latter focus on the behavior of the algorithm during the runtime, e.g., how fast solutions converge, how an algorithm behaves in case a specific technique or parameter is used, etc. For most methods focusing on solving a specific problem in a defined domain, optimality-based measures are a reasonable choice as usually the final solution is the most important quality indicator.

However, as the underlying monitor selection problem is applied to a highly-dynamic search space, optimality-based and behavior-based measures start to converge as soon as each solution at an arbitrary stage during the optimization process becomes relevant for deployment in the actual network. Hence, here the main focus is on optimality-based measures for determining the quality of the optimization method using normalized scoring as proposed in [Nguyen et al., 2012] while additional information is provided using time-series plots for a selected set of representative problem instances.

As the fitness values of two consecutive results are not comparable due to the highly-dynamic search space (the amount of active nodes and edges in the model differs for two different candidate solutions), normalization provides a robust way to address this issue. In this case, the fitness value is again divided into both initial parts: number of monitors and edge coverage. The values are normalized to the $[0, 1]$ interval, such that the following ratios apply:

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

$$edge\ coverage_t = 1 - \left(\frac{\sum\limits_{\{v_i,v_j\} \in S_t} w(v_i, v_j)}{\sum\limits_{\{v_i,v_j\} \in E_t} w(v_i, v_j)} \right) \tag{7}$$

where $S_t = \{\{v_i, v_j\} \in E_t \mid x_i = 0 \wedge x_j = 0\}$ of network model graph $G_t = (V_t, E_t, w)$ 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 edge coverage while minimizing the used monitors. Displaying the individual parts of the fitness function is only used for evaluation of results. The optimization in our experiments remains single-objective using the scalarized fitness value for the optimization process as having uncovered edges is not an option for deployment in an active network. While a formulation as bi-objective optimization problem is possible, we decided to describe the problem under study as single-objective one for aforementioned reasons. A formulation as multi-objective problem and according optimization is to be evaluated separately as future work.

Tables 3, 4, and 5 provide numerical results for a representative set of 5 different problem instances for each of the methods under study: the common EA, the BMS LS, and the hybrid LS EA respectively. Here, each row shows the results of the median and its IQR (columns MEDIAN and IQR) of the used monitors (column USED MONITORS) and the edge coverage (column EDGE COVERAGE) for the respective problem instance (column INSTANCE) and volatility level (column VOLATILITY). For the sake of brevity and overview, detailed results for all problem instances are omitted. However, the last three rows show the median value of the median and IQR over all 39 problem instances for the amount of used monitors and the edge coverage for each volatility level. As the results have been found to be non-normally distributed using appropriate statistical methods, the median and corresponding IQR have been used instead of mean and SD.

Another interesting measure for the optimization is wall-clock runtime, which is captured for all experiments and shown in Table 6. Here, each row provides information about the mean runtime (column MEAN) and SD (column SD), both in seconds, for the optimization for each combination of instance (column INSTANCE) and volatility level (column VOLATILITY) and optimization method (columns COMMON EA, LS EA, and BMS LS). Again, the last three rows show the median value of the mean and SD over all 39 problem instances for the runtime of each of the three volatility levels.

In order to answer the research questions and derived hypotheses provided, Table 7 reports the p-values and statistics for the five representative problem instances for comparing the BMS LS with the common EA (column EA VS. BMS LS), the LS EA with the BMS LS (column LS EA VS. BMS LS), and the LS EA with the common EA (column LS EA VS. EA). As results have been found to be non-normally distributed, the one-sided Mann-Whitney U-test [Mann and Whitney, 1947] has been applied to determine the statistical values. In addition to the set of selected problem instances, Table 8 provides a summary of the statistical results for all of the 39 problem instances for each volatility level and experiment. Here, the percentage gives the relative amount of instances where $p \leq 0.05$, i.e., where sufficient evidence is found to reject the proposed hypothesis.

In order to provide information about the optimization behavior of the individual optimization methods for each volatility level, Figure 1 provides plots for a selected set of four different representative problem instances. The different problem instances are shown column-wise (`NREN`, `internet-as`, `p2p-Gnutella31`, and `delaunay_n11`), while the three volatility levels are given row-wise from 0.05 over 0.10 to 0.25. In each plot, the x-axis shows the number of evaluations from 0 to the maximum number of evaluations (100 000), while the y-axis shows

Table 3: Experimental results of the optimization analysis for the common EA results on highly dynamic communication networks. The last three rows show the median value of the median and IQR over all problem instances.

		Used Monitors		Edge Coverage	
Instance	Volatility	Median	IQR	Median	IQR
NREN	5	0.43	0.01	1.00	0.00
NREN	10	0.46	0.01	1.00	0.00
NREN	25	0.52	0.02	1.00	0.00
internet-as	5	0.54	0.00	0.97	0.00
internet-as	10	0.53	0.00	0.96	0.00
internet-as	25	0.51	0.00	0.88	0.01
p2p-Gnutella31	5	0.58	0.00	0.84	0.01
p2p-Gnutella31	10	0.57	0.00	0.81	0.01
p2p-Gnutella31	25	0.55	0.00	0.74	0.01
delaunay_n11	5	0.78	0.01	1.00	0.00
delaunay_n11	10	0.79	0.01	1.00	0.00
delaunay_n11	25	0.80	0.01	0.98	0.00
frb40-19-4	5	0.99	0.00	1.00	0.00
frb40-19-4	10	0.99	0.00	1.00	0.00
frb40-19-4	25	0.98	0.01	1.00	0.00
All instances	5	0.72	0.01	1.00	0.00
All instances	10	0.70	0.01	0.99	0.00
All instances	25	0.65	0.01	0.97	0.00

the ratio of the individual measure and method on a scale from 0.25 to 1.00. The measure for the edge coverage is plotted in dotted lines and the measure for used monitors in solid lines. Each method is shown by an individual color, where blue is used for showing both measures for the common EA, yellow for the LS EA, and green for the BMS LS, respectively.

6.2 Discussion

As can be seen in Tables 3, 4, and 5, both the common EA and hybrid LS EA perform significantly better than the BMS LS approach. While the median edge coverage for all instances are $\geq 97\%$ for the experiments using the common EA and the hybrid LS EA, the median edge coverage for the BMS LS is only 59% for volatility levels 0.05 and 0.10 or 57% for a volatility level of 0.25. The IQRs for

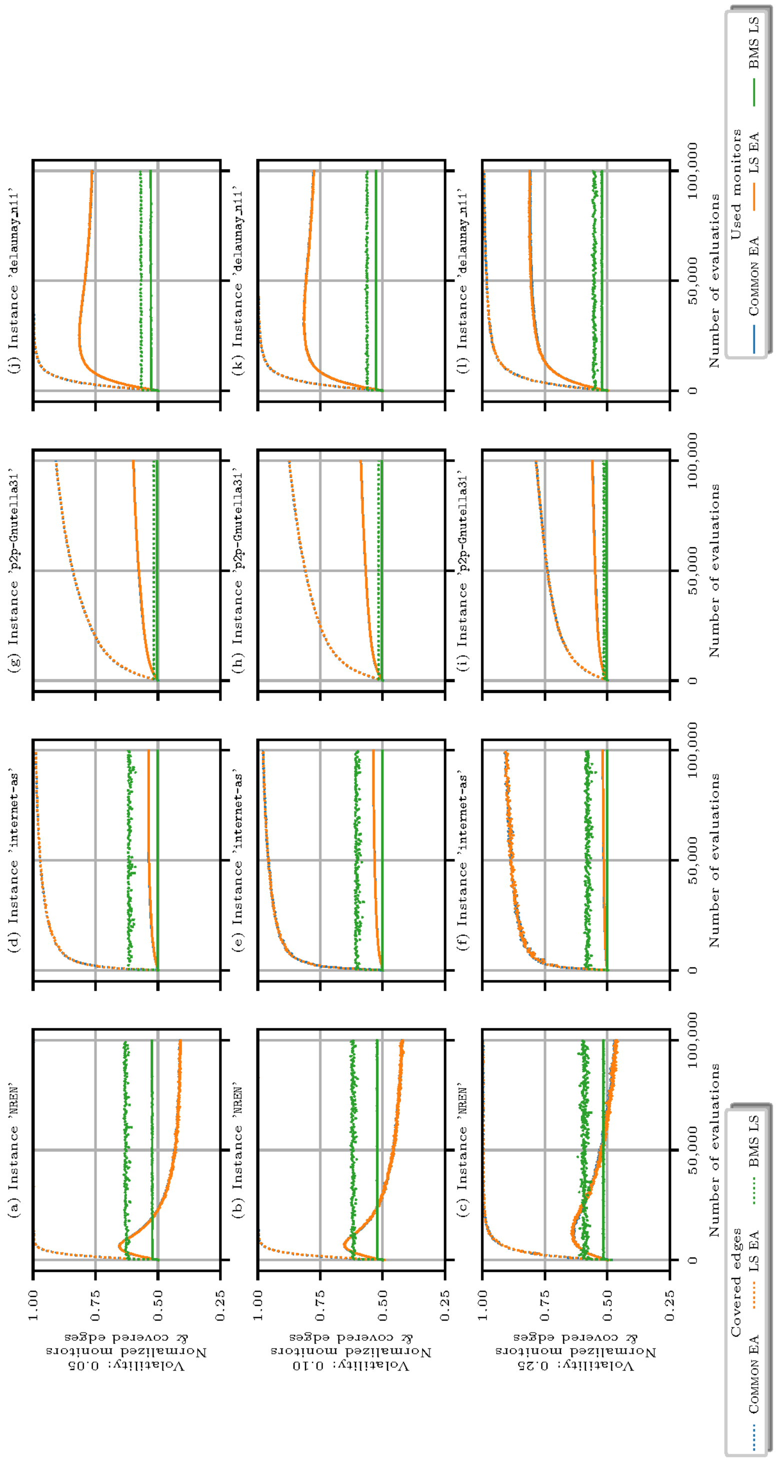

Figure 1: Runtime behavior plots for EA runs on different problem instances.

Table 4: Experimental results of the optimization analysis for the BMS LS results on highly dynamic communication networks. The last three rows show the median value of the median and IQR over all problem instances.

		Used Monitors		Edge Coverage	
Instance	Volatility	Median	IQR	Median	IQR
`NREN`	5	0.52	0.02	0.63	0.04
`NREN`	10	0.52	0.02	0.62	0.04
`NREN`	25	0.52	0.02	0.59	0.05
`internet-as`	5	0.50	0.00	0.62	0.02
`internet-as`	10	0.50	0.00	0.61	0.03
`internet-as`	25	0.50	0.00	0.58	0.04
`p2p-Gnutella31`	5	0.50	0.00	0.52	0.00
`p2p-Gnutella31`	10	0.50	0.00	0.52	0.00
`p2p-Gnutella31`	25	0.50	0.00	0.51	0.01
`delaunay_n11`	5	0.53	0.01	0.57	0.01
`delaunay_n11`	10	0.53	0.01	0.56	0.02
`delaunay_n11`	25	0.52	0.01	0.55	0.02
`frb40-19-4`	5	0.57	0.03	0.63	0.05
`frb40-19-4`	10	0.56	0.01	0.61	0.02
`frb40-19-4`	25	0.55	0.01	0.59	0.03
`All instances`	5	0.52	0.01	0.59	0.02
`All instances`	10	0.51	0.01	0.59	0.02
`All instances`	25	0.51	0.01	0.57	0.02

all experiments for both, the used monitors and edge coverage, indicate that the confidence of the median value is high (≤ 0.05). Regarding the used monitors it can be observed that the median value of this measure is higher for the common EA and the LS EA (65% to 72%) than for the BMS LS method (51% to 52%). As a higher coverage of edges is achieved, it is possibly necessary to also use a higher amount of monitors, which is why a higher amount of monitors does, at a first glance, not necessarily indicate a lower quality.

Indeed, evaluating the individual instances, it can be observed that some experiments require nearly the same amount of monitors while maintaining a significantly higher coverage of the network. This can be seen, e.g., in the results of problem instance `internet-as` at volatility level 0.25, where the common EA and the LS EA both require 51% of the nodes as monitors to cover 88% of the edges, while the BMS LS method requires 50% of the nodes to monitor only 58%

Table 5: Experimental results of the optimization analysis for the LS EA results on highly dynamic communication networks. The last three rows show the median value of the median and IQR over all problem instances.

Instance	Volatility	Used Monitors Median	Used Monitors IQR	Edge Coverage Median	Edge Coverage IQR
NREN	5	0.43	0.01	1.00	0.00
NREN	10	0.46	0.01	1.00	0.00
NREN	25	0.52	0.02	1.00	0.00
internet-as	5	0.54	0.00	0.97	0.00
internet-as	10	0.53	0.00	0.96	0.00
internet-as	25	0.51	0.00	0.88	0.01
p2p-Gnutella31	5	0.58	0.00	0.84	0.00
p2p-Gnutella31	10	0.57	0.00	0.81	0.01
p2p-Gnutella31	25	0.55	0.00	0.74	0.01
delaunay_n11	5	0.78	0.01	1.00	0.00
delaunay_n11	10	0.79	0.01	1.00	0.00
delaunay_n11	25	0.80	0.01	0.98	0.00
frb40-19-4	5	0.99	0.01	1.00	0.00
frb40-19-4	10	0.99	0.01	1.00	0.00
frb40-19-4	25	0.99	0.01	1.00	0.00
All instances	5	0.72	0.00	1.00	0.00
All instances	10	0.69	0.01	0.99	0.00
All instances	25	0.65	0.01	0.97	0.00

of the whole network.

In the experiments conducted, it can be observed that in case the volatility level is higher, the edge coverage and used monitors are both reduced. However, it can also be observed that the size of the problem instance, i.e., the number of nodes and links of the network, has a much higher impact on result quality than the volatility level. For most of the small instances, e.g., `NREN` or `delaunay_n11`, an (almost) full coverage is achieved, while most of the larger instances leave room for further improvement, e.g., `p2p-Gnutella31`. While the latter is obvious due to an increase of the search space of the problem, it is a positive observation that the proposed search heuristics show such a level of robustness to counteract changes in the dynamic problem instances.

The aforementioned behavior is substantiated by the runtime behavior plots in Figure 1. For the experiments using the common EA or LS EA, it can be

Table 6: Experimental results of the wall-clock runtime analysis (in seconds) for the optimization for all methods under study on highly dynamic communication networks. The last three rows show the median value of the median and IQR over all problem instances.

		Common EA		LS EA		BMS LS	
Instance	Volatility	Mean	SD	Mean	SD	Mean	SD
NREN	5	4.87	1.44	4.72	1.30	5.82	2.13
NREN	10	4.75	1.27	5.06	1.50	5.84	2.05
NREN	25	5.62	1.40	5.44	1.40	6.44	1.77
internet-as	5	460.11	157.70	486.18	176.14	575.85	258.34
internet-as	10	616.30	233.37	624.62	487.13	819.18	649.40
internet-as	25	1048.75	517.40	1070.56	707.96	1017.95	823.71
p2p-Gnutella31	5	979.97	584.36	859.90	572.59	954.91	500.56
p2p-Gnutella31	10	1118.57	423.42	1229.79	758.71	1371.82	824.94
p2p-Gnutella31	25	2004.57	1059.25	1978.52	1205.57	1854.74	1022.97
delaunay_n11	5	12.99	3.48	13.67	3.99	18.03	5.66
delaunay_n11	10	13.17	3.73	13.23	3.42	18.56	6.52
delaunay_n11	25	14.19	3.60	14.49	4.01	18.02	7.22
frb40-19-4	5	898.80	324.07	1083.00	713.02	848.46	523.49
frb40-19-4	10	1310.37	829.11	1236.65	698.32	1080.71	554.48
frb40-19-4	25	1535.09	1056.41	1512.38	825.64	1281.61	839.29
All instances	5	292.08	171.30	281.80	158.70	349.38	142.38
All instances	10	379.45	230.87	454.93	256.07	447.63	290.47
All instances	25	622.89	453.97	544.44	457.27	675.65	399.82

observed that smaller instances as NREN reach a state of (almost) full coverage early in the search process (after around 10% of the total number of evaluations). After reaching a sufficiently high coverage, the optimization starts to find solutions with a lower amount of monitors while maintaining the coverage as indicates in the plots. The same behavior also holds for runtime plots of instance delaunay_n11. The volatility level influences the search by slightly reducing optimization speed, as indicated by the plots. However, in case of larger instances, full coverage is possibly not reached yet, but the runtime behaviour plot indicates a steady improvement over the whole runtime. Hence, providing a higher number of evaluations is supposed to lead to further improvement of the results.

An important factor for practical applicability of reactive network optimiza-

Table 7: Statistics and p-values for testing the optimization hypotheses $H1_1$, $H2_1$, and $H3_1$.

		EA vs. BMS LS		LS EA vs. BMS LS		LS EA vs. EA	
Instance	Volat.	p-value	Statistic	p-value	Statistic	p-value	Statistic
NREN	5	$\ll 0.01$	10 000.0	$\ll 0.01$	10 000.0	0.38	5 122.5
NREN	10	$\ll 0.01$	10 000.0	$\ll 0.01$	10 000.0	0.38	5 122.5
NREN	25	$\ll 0.01$	10 000.0	$\ll 0.01$	10 000.0	0.38	5 122.5
internet-as	5	$\ll 0.01$	10 000.0	$\ll 0.01$	10 000.0	0.82	4 619.0
internet-as	10	$\ll 0.01$	10 000.0	$\ll 0.01$	10 000.0	0.82	4 619.0
internet-as	25	$\ll 0.01$	10 000.0	$\ll 0.01$	10 000.0	0.82	4 619.0
p2p-Gnutella31	5	$\ll 0.01$	10 000.0	$\ll 0.01$	10 000.0	0.50	4 998.5
p2p-Gnutella31	10	$\ll 0.01$	10 000.0	$\ll 0.01$	10 000.0	0.50	4 998.5
p2p-Gnutella31	25	$\ll 0.01$	10 000.0	$\ll 0.01$	10 000.0	0.50	4 998.5
delaunay_n11	5	$\ll 0.01$	10 000.0	$\ll 0.01$	10 000.0	0.64	4 853.5
delaunay_n11	10	$\ll 0.01$	10 000.0	$\ll 0.01$	10 000.0	0.64	4 853.5
delaunay_n11	25	$\ll 0.01$	10 000.0	$\ll 0.01$	10 000.0	0.64	4 853.5
frb40-19-4	5	$\ll 0.01$	10 000.0	$\ll 0.01$	10 000.0	0.03	5 796.0
frb40-19-4	10	$\ll 0.01$	10 000.0	$\ll 0.01$	10 000.0	0.03	5 796.0
frb40-19-4	25	$\ll 0.01$	10 000.0	$\ll 0.01$	10 000.0	0.03	5 796.0

Table 8: Summary for statistical analysis over all problem instances. Shows percentage of problem instances where $p \leq 0.05$ per volatility level.

Instance	Volatility	EA vs. BMS LS	LS EA vs. BMS LS	LS EA vs. EA
All instances	5	100%	100%	17.95%
All instances	10	100%	100%	17.95%
All instances	25	100%	100%	17.95%

tion is the absolute wall-clock runtime of the optimization. As these values are highly dependent on the implementation and scales with the hardware used, it should be considered a rough estimate and as comparison between different problem instance attributes and optimization method parameters. As can be seen in Table 6, runtime of individual experiments having 100 000 evaluations is between several seconds for smaller instances up to several minutes for larger ones. One significant factor increasing runtime is the volatility level. However, as the experiments are carried out generating the different models on the fly during the optimization process, this behavior might be implementation-related and may be reduced in case another implementation is used, the models are precomputed or the information about changes in the network is gathered differently. Regarding all instances, there seems to be no major difference between the different optimization methods. Hence, in case of a volatility level of 0.05, it is possible to do ≈ 325 evaluations per second while having a volatility level of

0.25 decreases this rate to ≈ 194 evaluations per second.

With respect to the comparison of the individual methods, the statistical analysis as shown in Tables 7 and 8 offer meaningful results. Comparing the common EA with the BMS LS, the statistics offer the unambiguous results that the common EA outperforms the BMS LS on all problem instances. Therefore, null hypothesis $H1_0$ can be rejected accepting hypothesis $H1_1$: yes, the median fitness of the best solutions is improved using an EA for optimization of highly-dynamic communication networks as opposed to using a BMS LS heuristic. Hence, research question **RQ2** can be answered: yes, the proposed EA does improve the solutions compared to the common BMS LS in the optimization of highly-dynamic communication networks.

The same holds for the comparison between the LS EA and the BMS LS methods. Here, the hybrid LS EA offers significantly better results on all problem instances under study. There is also sufficient evidence to reject null hypothesis $H2_0$ and thus accept $H2_1$: yes, the median fitness of the best solutions is improved when a LS EA is used for optimization of highly-dynamic communication networks as opposed to using a BMS LS heuristic. This leads to the answer of research question **RQ3**: yes, the LS EA does improve the solutions compared to the common BMS LS in the optimization of highly-dynamic communication networks.

Regarding the last hypothesis, $H3_1$, results of the comparison of the common EA and the hybrid LS EA have to be evaluated. According to Table 8, only in 17% of the problem instances under study, the LS EA generated significantly better results than the common EA. As this is not sufficient to reject $H3_0$ generally, the hypothesis $H3_1$ must also be rejected: no, the median fitness of the best solutions is not improved when the LS EA is used for optimization of highly-dynamic communication networks as opposed to using the EA. As for the hypothesis, the related research question **RQ4** must also be declined: no, the proposed LS EA does not necessarily improve the solutions compared to the EA in the optimization of highly-dynamic communication networks.

However, even though there is no clear distinction between a pure EA and a hybrid LS EA possible, both methods offer promising results for the reactive heuristic optimization of highly-dynamic communication infrastructures as shown by the results of the experiments conducted. Therefore, the general research question **RQ1** can be answered: yes, it is possible to optimize highly-dynamic communication network infrastructures using the proposed heuristic search.

6.3 A Note on Complexity

Unfortunately, the tractability of this problem in the general case, even for the simplest MVC variant, is an open problem. All we can offer is a crude worst-case estimation for the computational problem involved, based on Complexity Theory. As with any other $\mathcal{NP}$-hard problem, the best we can say is that it seems to require superpolynomial time in the worst case. This is a general result that can be stated independently of the particular algorithm and implementation used, as long as $\mathcal{P}$ is not equal to $\mathcal{NP}$. In fact, no algorithm with subexponential worst-case time is known for any $\mathcal{NP}$-hard problem and, what is more, such algorithms cannot exist under the *Exponential Time Hypothesis (ETH)* [Impagliazzo and Paturi, 2001].

It is true that we are fortunate that algorithms for hard problems may behave much better in practice for small and even medium-size instances and, sometimes, for some large instances. Of course, according to Complexity Theory, $\mathcal{NP}$-hardness implies that this does not prevent the existence of pathological or ill-conditioned instances of even moderate size that can drive any exact algorithm to its exponential worst case. One could think that the situation should be better than this, as we are not in fact guaranteeing an exact solution, but trying to approximate the optimal solution. However, this is not the case. Unless, $\mathcal{P} = \mathcal{NP}$, MVC cannot be approximated within a factor of 1.3606 [Dinur and Safra, 2005]. In fact, the classical greedy constant-factor approximation algorithm for this problem can just guarantee that the vertex cover found is twice as large as an optimal one.

Therefore, to go further, at the expense of exactness and of even constant-factor approximation guarantees, we decided to resort to metaheuristics. The problem is that for elaborate metaheuristic algorithms as the ones we present here, good behavior is a subject of empirical analysis, as there are many parameters involved and a formal analysis of, e.g., genetic algorithms is not possible with the current state of the art, save for the simplest algorithms and functions [Droste et al., 2002, Jansen and Wegener, 2006, Oliveto and Witt, 2015]. According to Neumann and Witt [Neumann and Witt, 2010]:

> In recent years, a number of publications regarding stochastic search algorithms for the vertex cover problem have appeared. First, some simple evolutionary algorithms for single-objective optimization have been investigated on this problem. It is shown in Friedrich, Hebbinghaus, Neumann, He, and Witt (2007) that a natural single-objective approach which minimizes the number of vertices and penalizes the number of uncovered edges has an exponential optimization even on simple bipartite graphs. Additional negative results regarding the single-objective search algorithms were presented by Oliveto, He, and Yao (2009) and Oliveto, He, and Yao (2008), who show that the use of populations in single-objective formulations does not necessarily allow for a significant increase in success probability.

The aforementioned algorithms are the $(1 + 1)$ EA and the simplest *Ran-*

domized Local Search (RLS) algorithm. Most available theoretical results are restricted to these algorithms. As the algorithms that we propose are not so simple and the functions to optimize are rather complex, all that we can say is that, for a fixed size of the population, the number of evaluations is the main parameter through which the execution time can be controlled. Whichever the true complexity is, it increases linearly with the number of evaluations and, as we are interested in dynamic optimization, we work backwards: we fix a time window and try to do our best in the available time, i.e., in this case the number of evaluations.

However, thanks to extensive experimentation, we can show evidence that problems large enough to be of practical interest in our domain can be solved by our techniques. The authors provide 15 additional pages of data backing this statement, with experimental results showing that we can get reasonable approximations dynamically.[2]

7 Conclusion and Outlook

In this paper, the general applicability of heuristic methods for reactive optimization of highly-dynamic communication networks has been studied. Due to the increasing complexity and dynamics of modern communication networks, a steady optimization of communication networks becomes more and more inevitable for providing reliable communication infrastructures of high quality.

As described in more detail in the previous sections, a successful application of optimization heuristics in conjunction with the developing software-defined networking and network function virtualization standards enable a new way of automated and reactive network management. Here, several established methods for network optimization have been applied and compared scientifically, using the known $\mathcal{NP}$-hard optimization monitor selection problem for testing the general applicability of the method to an exemplary but important problem in network management. As the results show, it is possible and efficient to use EAs for optimization of communication networks. The main research question, whether it is possible to optimize highly-dynamic communication network infrastructures using a heuristic search method, could be confirmed based on comprehensive experiments on a variety of 39 different real-world and synthetic benchmark problem instances.

This research is a first step towards an effective, reactive, and automated network management and forms the base for possible interesting future aspects. Due to the time-criticality of the problem, experiments in a live environment of adequate size is one major aspect of future research, tuning the optimization methods according to the target environment. Furthermore, replacing the

[2] Please, find the data as raw CSV files and a PDF summary in https://github.com/rmuellerb/DynamicOptimizationData.

generation of random changes in the network with realistic changes obtained from the actual network using SDN or NFV and measuring the performance are other interesting aspects. An extensive comparison of further methods using, e.g., aforementioned NuMVC, FastVC or a different bio-inspired heuristic, could be interesting in order to find the most suitable heuristic for optimization of highly-dynamic communication networks and further improve the effectiveness of the presented approach.

Acknowledgements

We would like to take the chance to thank the reviewers for providing constructive and helpful feedback to enhance the quality of this paper. This work has been funded 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 project FAME (RTI2018-093608-B-C33). Responsible for the content are the authors.

References

[Atiqullah and Rao, 1993] Atiqullah, M. M. and Rao, S. (1993). Reliability optimization of communication networks using simulated annealing. *Microelectronics Reliability*, 33(9):1303–1319.

[Baccelli and Zuyev, 1999] Baccelli, F. and Zuyev, S. (1999). Poisson-Voronoi Spanning Trees with Applications to the Optimization of Communication Networks. *Operations Research*, 47(4):619–631.

[Beyer and Schwefel, 2002] Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution strategies - A comprehensive introduction. *Natural Computing*, 1(1):3–52.

[Branke and Schmeck, 2003] Branke, J. and Schmeck, H. (2003). *Designing Evolutionary Algorithms for Dynamic Optimization Problems*, pages 239–262. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Cai, 2015] Cai, S. (2015). Balance between complexity and quality: Local search for minimum vertex cover in massive graphs. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 747—-753.

[Cai et al., 2017] Cai, S., Lin, J., and Luo, C. (2017). Finding A Small Vertex Cover in Massive Sparse Graphs: Construct, Local Search, and Preprocess. *Journal of Artificial Intelligence Research*, 59:463–494.

[Chauhan et al., 2017] Chauhan, A., Friedrich, T., and Quinzan, F. (2017). Approximating optimization problems using EAs on scale-free networks. In *Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '17*, pages 235–242, New York, New York, USA. ACM Press.

[Chen et al., 2010] Chen, J., Kanj, I. A., and Xia, G. (2010). Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756.

[Chiang, 2009] Chiang, M. (2009). Nonconvex Optimization for Communication Networks. In *Advances in Applied Mathematics and Global Optimization. Advances in Mechanics and Mathematics*, pages 137–196.

[Coello Coello, 2012] Coello Coello, C. A. (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, page 849, New York, New York, USA. ACM Press.

[Cruz et al., 2011] Cruz, C., Gonzalez, J. R., and Pelta, D. A. (2011). Optimization in dynamic environments: a survey on problems, methods and measures. *Soft Computing*, 15(7):1427–1448.

[Dasgupta et al., 2012] Dasgupta, M., Biswas, G. P., and Bhar, C. (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)*, pages 256–262. IEEE.

[Diestel, 2017] Diestel, R. (2017). *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 5th edition.

[Dinur and Safra, 2005] Dinur, I. and Safra, S. (2005). On the hardness of approximating vertex cover. *Annals of Mathematics*, 162:439–485.

[Doerr et al., 2012] Doerr, B., Johannsen, D., and Winzen, C. (2012). Multiplicative Drift Analysis. *Algorithmica*, 64(4):673–697.

[Droste et al., 2002] Droste, S., Jansen, T., and Wegener, I. (2002). On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1):51–81.

[Gen and Cheng, 1999] Gen, M. and Cheng, R. (1999). *Genetic Algorithms and Engineering Optimization*, volume 7 of *Wiley Series in Engineering Design and Automation*. John Wiley & Sons, Inc., Hoboken, NJ, USA.

[Hamming, 1950] Hamming, R. W. (1950). Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, 29(2):147–160.

[Holland and Leinhardt, 1971] Holland, P. W. and Leinhardt, S. (1971). Transitivity in Structural Models of Small Groups. *Comparative Group Studies*, 2(2):107–124.

[Impagliazzo and Paturi, 2001] Impagliazzo, R. and Paturi, R. (2001). On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375.

[ITU, 2013] ITU (2013). Facts and figures. In *International Telecommunication Union*, pages 14–16. Geneva, Switzerland.

[Jansen and Wegener, 2006] Jansen, T. and Wegener, I. (2006). On the analysis of a dynamic evolutionary algorithm. *Journal of Discrete Algorithms*, 4(1):181–199.

[Karp, 1972] Karp, R. M. (1972). Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*, pages 85–103. Springer US, Boston, MA.

[Knight et al., 2012] Knight, S., Falkner, N., Nguyen, H. X., Tune, P., and Roughan, M. (2012). I can see for miles: Re-visualizing the internet. *IEEE Network*, 26(6):26–32.

[Lagraa and Francois, 2017] Lagraa, S. and Francois, J. (2017). Knowledge discovery of port scans from darknet. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 935–940. IEEE.

[Leskovec et al., 2007] Leskovec, J., Kleinberg, J., and Faloutsos, C. (2007). Graph evolution. *ACM Transactions on Knowledge Discovery from Data*, 1(1):2–es.

[Lopez-Ibanez et al., 2016] Lopez-Ibanez, M., Dubois-Lacoste, J., Perez Caceres, L., Birattari, M., and Stuetzle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

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

[Marotta et al., 2017] Marotta, A., D'Andreagiovanni, F., Kassler, A., and Zola, E. (2017). On the energy cost of robustness for green virtual network function placement in 5G virtualized infrastructures. *Computer Networks*, 125:64–75.

[Michalewicz et al., 2007] Michalewicz, Z., Schmidt, M., Michalewicz, M., and Chiriac, C. (2007). Adaptive Business Intelligence: Three Case Studies. In *Studies in Computational Intelligence*, volume 51, pages 179–196. Springer, Berlin, Heidelberg.

[Mueller-Bady et al., 2017a] Mueller-Bady, R., Kappes, M., Atkinson, L., and Medina-Bulo, I. (2017a). Multijob. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '17*, pages 1231–1238, New York, New York, USA. ACM Press.

[Mueller-Bady et al., 2017b] Mueller-Bady, R., Kappes, M., Medina-Bulo, I., and Palomo-Lozano, F. (2017b). Optimization of monitoring in dynamic communication networks using a hybrid evolutionary algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '17*, pages 1200–1207, New York, New York, USA. ACM Press.

[Mueller-Bady et al., 2018] Mueller-Bady, R., Kappes, M., Medina-Bulo, I., and Palomo-Lozano, F. (2018). Using evolutionary dynamic optimization for monitor selection in highly dynamic communication infrastructures. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '18*, pages 1672–1679, New York, New York, USA. ACM Press.

[Neri and Maekinen, 2007] Neri, F. and Maekinen, R. A. E. (2007). Hierarchical Evolutionary Algorithms and Noise Compensation via Adaptation. In *Studies in Computational Intelligence*, volume 51, pages 345–369. Springer.

[Neumann and Witt, 2010] Neumann, F. and Witt, C. (2010). *Bioinspired Computation in Combinatorial Optimization. Algorithms and Their Computational Complexity.* Springer.

[Newman, 2003] Newman, M. E. J. (2003). Mixing patterns in networks. *Physical Review E*, 67(2):026126.

[Nguyen et al., 2012] Nguyen, T. T., Yang, S., and Branke, J. (2012). Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, 6:1–24.

[Oliveto and Witt, 2015] Oliveto, P. S. and Witt, C. (2015). Improved time complexity analysis of the simple genetic algorithm. *Theoretical Computer Science*, 605:21–41.

[Quintao et al., 2005] Quintao, F., Nakamura, F., and Mateus, G. (2005). Evolutionary Algorithm for the Dynamic Coverage Problem Applied to Wireless Sensor Networks Design. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1589–1596. IEEE.

[Ripeanu and Foster, 2002] Ripeanu, M. and Foster, I. (2002). Mapping the Gnutella Network: Macroscopic Properties of Large-Scale Peer-to-Peer Systems. In *IEEE Internet Computing Journal*, pages 85–93.

[Rohlfshagen and Yao, 2008] Rohlfshagen, P. and Yao, X. (2008). Attributes of Dynamic Combinatorial Optimisation. In Li, X., Kirley, M., Zhang, M., Green, D., Ciesielski, V., Abbass, H., Michalewicz, Z., Hendtlass, T., Deb, K., Tan, K. C., Branke, J., and Shi, Y., editors, *Simulated Evolution and Learning*, pages 442–451, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Rossi and Ahmed, 2015] Rossi, R. A. and Ahmed, N. K. (2015). The Network Data Repository with Interactive Graph Analytics and Visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 4292–4293. AAAI Press.

[Shengxiang Yang et al., 2010] 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):52–63.

[Tange, 2011] Tange, O. (2011). GNU Parallel: the command-line power tool. *The USENIX Magazine*, 36(1):42–47.

[Yang et al., 2008] Yang, K., Wu, Y., Huang, J., Wang, X., and Verdu, S. (2008). Distributed Robust Optimization for Communication Networks. In *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, pages 1157–1165. IEEE.