Multi-objective multi-constrained QoS Routing in large-scale networks: A genetic algorithm approach

Quang Tran Anh Pham, Jean-Michel Sanner, Cedric Morin, Yassine Hadjadj-Aoul

Multi-objective multi-constrained QoS Routing in large-scale networks: A genetic algorithm approach

Pham Tran Anh Quang*, Jean-Michel Sanner†, Cédric Morin*, Yassine Hadjadj-Aoul*
*b-com, 1219 avenue des Champs Blancs - 35510 Cesson-Sévigné - France
Email:{quang.pham-tran-anh,cedric.morin, yhadjadj}@b-com.com †Orange Labs R&D, Rennes
Email:jeanmichel.sanner@orange.com

***Abstract*—The growing need for a simplified management of network infrastructures has recently led to the emergence of software-defined networking (SDN), which enables a centralized path calculation. The diversification of services, as well as the need of rapid path deployment, raises, however, challenges in routing algorithms. Moreover, Quality of Service (QoS) requirements and conflicts between them pile up the complexity of the problem. An intuitive method is formulating the problem as an Integer Linear Programming and solving it by an approximation algorithm. This method tends to have a specific design and usually suffers from unacceptable computational delays to provide a sub-optimal solution. Genetic algorithms (GAs) are deemed as a promising solution to cope with highly complex optimization problems. However, the convergence speed and the quality of solutions should be addressed in order to fit into practical implementations. In this paper, we propose a genetic algorithm-based mechanism to address the multi-constrained multi-objective routing problem. Using a repairer to reduce the search space to feasible solutions, results confirm that the proposed mechanism is able to find the Pareto-optimal solutions within a short run-time.**

I. INTRODUCTION

In recent decades, we have been witnessing an incredibly fast development of the Internet in both its coverage and traffic load. It unveils dramatic challenges in optimizing and guaranteeing QoS for large-scale networks. One of major components that contributes significantly to the performance of large-scale networks is routing. Routing algorithm is in the core of routing protocol and is responsible for determining the optimal or sub-optimal paths between a source and a destination. A QoS routing algorithm takes QoS and network constraints into account in order to find the optimal path in terms of predefined objectives.

QoS routing plays an important role in QoS provisioning in networks. Depending on the QoS requirements, different QoS routing problems can be defined. Generally, they could be classified into the following five types:

- *Shortest path (SP):* The route minimizes a unique end-to-end metric
- *Constrained Shortest Path (CSP):* The route minimizes an end-to-end metric while maintaining another metric below a given bound.
- *Multi-constrained shortest path (MCSP):* CSP proble m with several end-to-end metrics constrained by prescribed bounds.
- *Multi-constrained path (MCP):* MCSP problem without end-to-end metric optimization.
- *Multi-constrained Multi-objective path (MCMOP):* MCP problem with multiple end-to-end metrics to optimize.

Both MCP and MCSP are NP-complete if the metrics are mutually independent [1]. MCSP is considered as a modified version of MCP problem and it is more difficult than MCP problem. The solution of MCSP problem is also a solution to the MCP problem, but not vice versa [2]. MCSP is a special case of MCMOP when there is only one objective, thus MCMOP is also an NP-complete problem. Artificial Intelligent (AI) techniques, e.g. Genetic algorithms (GAs) [3], are able to deal with high complexity of routing problems. However, existing genetic algorithms have typically fairly high execution time, thus not well suited for on-line routing decisions [4].

In this paper, we propose a genetic algorithm that can cope with the high complexity of MCMOP routing problems even in large-scale networks. The rest of this paper is structured as follows. Existing works related to this paper is discussed in Sec. II. A system overview is provided in Sec. III. Sec. IV describes a network model. The proposed genetic algorithm to address MCMOP is presented in Sec. V. Sec. VI provides the performance assessment of the proposed scheme. Finally, conclusions and future research are presented in Sec. VII.

II. RELATED WORKS

A. QoS Routing

Several approaches have been proposed in the literature to solve the problem of QoS routing. They could be classified into two main groups: (i) heuristic and (ii) exact algorithm. The authors in [5] have proposed a tunable accuracy multiple constraints routing algorithm (TAMCRA). It is built upon three fundamental concepts: a non-linear measure of the path length, the k-shortest path approach, and the principle of non-dominated paths. The non-linear measure of path length tends to perform well when the path weights are not correlated [2]. TAMCRA keeps track of up to k-non dominated path for each intermediate node i on the path from a source to a destination. Self-Adaptive Multiple Constraints Routing Algorithm (SAMCRA) [6] is an exact variant of TAMCRA. It differs from TAMCRA at the point that it could adapt the number of stored paths at each node. As an exact algorithm, SAMCRA assures that a feasible path is found if one exists.

B. Multi-objective Optimization Genetic Algorithm

Evolutionary algorithms have been exploited to cope with multi-objective optimization problem in [7], [8]. Although they

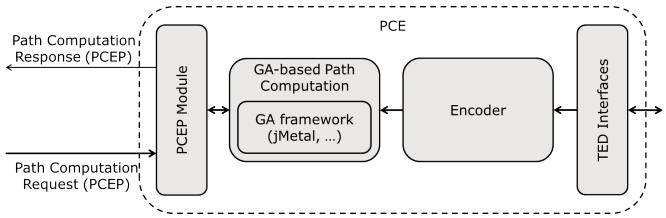

Fig. 1: Proposed GA-based path computation element

have proved their abilities in finding multiple non-dominated solutions on many test problems, it is necessary to introduce evolutionary algorithms that have better convergence speed. Elitism is a process that copies the fittest individuals to the next generation. By doing that, the solution quality will not degrade in the next generation and it helps in achieving better convergence [9]. The most well-known elitist multi-objective evolutionary algorithm (MOEAs) is Non-dominated Sorting Genetic Algorithm II (NSGA-II) [10] which outperforms other elitist MOEAs (Pareto Archived Evolution Strategy (PAES) and Strength Pareto Evolutionary Algorithm (SPEA)). However, a number of MOEAs is unable to find well-converged and diversified non-dominated solutions because of the loss of selection pressure in fitness evaluation [11] in problems with many objectives (more than 3 objectives). Unlike NSGA-II, NSGA-III selects the new population based on supplied reference points. NSGA-III have been demonstrated to work well for many-objective problems [12].

III. System Overview

The Path Computation Element (PCE) is the module which implements the route computation algorithm. It is a well documented element [13], that is already used in Multi Protocol Label Switching (MPLS) networks, for example. PCE communicates with its client through the Path Computation Element Protocol (PCEP) [14]. We chose to implement it out of the SDN controller in order to enforce simplicity, encapsulation, security and maintainability. It can implement any kind of routing algorithm, such as Dijkstra or, in our case, a genetic algorithm. We based our implementation on a modified version of the Netphony PCE [15]. Upon receiving a path computation request, the algorithm uses the network image provided by the Traffic Engineering Database (TED) to compute a result. Once this result is found, the PCE modifies the availability of the resources of the TED accordingly. For example, if a path that requests a bandwidth of 1 *Mbps* is sent through a link *A* with 3 *Mbps* available, then the link *A* will only propose 2 *Mbps* for the following computations. This strategy allows us to avoid allocating resources multiple time, which may lead to QoS violations. PCE's implementation details are presented in Fig. 1.

IV. Network model

We model a network as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of directed links. We introduce the binary variable Φ_e to index the usage of a directed link e. QoS measures of a link are presented by a vector $\{w_{i,e}, i = 1, ..., M\}$, where M is the number of QoS measures. End-to-end QoS measure i of a path is limited by an upper-bound W_i. QoS measures can be classified into three different classes: additive, multiplicative, and min-max QoS. Multiplicative measures are able to convert to additive measures by taking the logarithm [16]. For min-max QoS measures, the length of a path is the minimum or maximum of the QoS measures of links forming that path. A filter could be adopted to prune the network by removing the unqualified link. For instance, a video streaming requests a bandwidth of $5Mbps$, thus a filter will remove all links with capacity less than $5Mbps$. Meanwhile, path computation with additive routing is more difficult to cope with and it is proved to be NP-complete [17]. Consequently, in this paper we consider the cases where the length of a path is composed of additive QoS measures. We have $\sum_e \Phi_e w_{i,e} \leq W_i, \forall i$. The objectives are to minimize all QoS measures $\min \sum_e \Phi_e w_{i,e}$, for $i = 1 \dots M$. The cost of a path $l(\mathbf{P})$ is a M-dimension vector defined as follow $l(\mathbf{P}) = [l_1(\mathbf{P}), l_2(\mathbf{P}), \dots, l_M(\mathbf{P})]$, where $l_i(\mathbf{P}) = \sum_e \Phi_e w_{i,e}$. The objective is to find the set of non-dominated solutions (i.e. Pareto optimal).

The aforementioned mathematic model could be solved by the solvers that support multi-objective optimization such as Gurobi [18]. However, the high computational complexity of the problem, especially of large scale networks, causes a high runtime. In the next section, we re-encode this problem in order to solve it using genetic algorithms.

V. Genetic algorithm

In this paper, extra processes are added into the genetic algorithm in order to guarantee the feasibility of the population and shorten calculation time. Fig. 2 presents the proposed genetic algorithm flowchart. A repairer and a new crossover are added into the genetic algorithm. The set of candidate solutions is called as population. Each candidate solution (individual) is a sequence of bits representing Φ_e. In each iteration (generation), the quality of the population could be improved by executing crossover process and/or mutation process. Then, selected parents create the new offspring for the next generation. The following subsections describe the key components of the proposed scheme in details.

A. Initialization

To generate the initial population, links will be selected randomly among all available links. Consequently, they may not be feasible solutions. We decide to work on a search space composed of only feasible solutions, thus it is necessary to correct initial population. A repairer process presented in the next subsection addresses this need.

B. Repairer

This section discusses the proposed repairer, which is able to convert an unfeasible individual to a feasible individual. The repairing process aims to find non-dominated solutions.

Due to multiple QoS measures, a cost of a path is a vector composed of elements corresponding to QoS measures. The Dijkstra shortest path from n to d regarding QoS measure i is

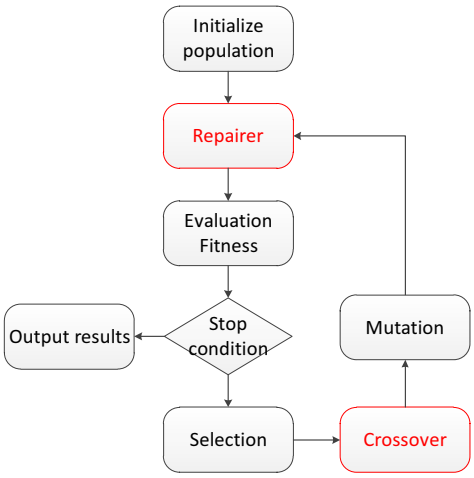

Fig. 2: Proposed Genetic Algorithm Flowchart

denoted as $p_i^{n,d}$ and its cost can be expressed as a vector $c_i^{n,d} = \left[c_{i,1}^{n,d}, ..., c_{i,M}^{n,d}\right]$, where M is the number of QoS measures and $c_{i,j}^{n,d}$ is the cost of QoS measure j. Following the definition of $c_i^{n,d}$, $c_{i,i}^{n,d} \leq c_{j,i}^{n,d}, \forall j \neq i$. The upper-bound of QoS measure i of the paths from n to d, denoted as $u_i^{n,d}$, is defined as $u_i^{n,d} = \max_{j \neq i} c_{j,i}^{n,d}$. The cost of a link from n to n' is denoted as a vector $c^{n,n'} = \left[c_1^{n,n'}, \ldots, c_M^{n,n'}\right]$, where $c_i^{n,n'}$ is the QoS measure i of link nn'.

Lemma 1. *If $c_{i,i}^{n',d} + c_i^{n,n'} > u_i^{n,d}$, there does not exist non-dominated path comprising link nn'.*

Proof. Assume that there is a non-dominated path $\mathcal{P}^*$ comprising link nn' and $c_{i,i}^{n',d} + c_i^{n,n'} > u_i^{n,d}, \forall i$. Intuitively, it exists a path from n to d, $\mathcal{P}_0$, such that $c_i^{\mathcal{P}_0} \leq u_i^{n,d}, \forall i$ (e.g. any shortest path from n to d). All paths to d consisting of link nn' have the QoS measure i greater than or equal to $c_{i,i}^{n',d} + c_i^{n,n'}$ since $c_{i,i}^{n',d}$ is the shortest length in QoS measure i. Consequently, all paths traversing through nn' are dominated by $\mathcal{P}_0$. Therefore, $\mathcal{P}^*$ could not be a non-dominated path. □

Lemma 2. *With additive QoS measures, all Pareto-optimal paths are simple paths*

Proof. A simple path is a path in a graph which does not have repeating vertices. Assume that there is a Pareto-optimal path $\mathcal{P}^*$ that is not a simple path with a vertex M_0 visited twice. s and d are the source and the destination, respectively. Let us denote the links originated from M_0 in first and second visits as (M_0, M_1) and (M_0, M_2), respectively. Due to additive metrics, we have the cost of $\mathcal{P}^*$, c^{P^*}, can be determined as the sum of sub-paths (all are simple paths), i.e. $c^{\mathcal{P}^*} = c^{\mathcal{P}_{s \to M_0}} + c^{\mathcal{P}_{M_0 \to M_0}} + c^{\mathcal{P}_{M_0 \to d}}$, where $\mathcal{P}_{s \to M_0}$ and $\mathcal{P}_{M_0 \to d}$ are the sub-paths from s to M_0 and from M_0 to d (traversing through M_2). $\mathcal{P}_{M_0 \to M_0}$ is the cost of the loop that traversing through $M_0, M_1, ..., M_0$. $\mathcal{P}^*$ is dominated by the simple path $\mathcal{P}'$ that has the same sub-paths from s to M_0 and from M_0 to d since the total cost of $\mathcal{P}'$ is $c^{\mathcal{P}_{s \to M_0}} + c^{\mathcal{P}_{M_0 \to d}}$ which is less than $c^{\mathcal{P}^*}$. Thus, $\mathcal{P}'$ dominates $\mathcal{P}^*$. It contradicts the assumption, thus all Pareto-optimal paths are simple paths. □

Alg. 1 describes the proposed repairer. The input of the repairer could be unfeasible. All active entries in the inputs are stored in the set $\tilde{\mathcal{E}}$ (line 3) and have higher priority to be selected through repairing procedure. For each QoS measure, the repairer finds the shortest path using Dijkstra algorithm (line 4), which is exploited to predict the cost of paths and eliminate dominated paths in later steps. The main loop of repairing process, from line 6 to line 22, in which all relaying nodes from the source to the destination are determined respectively. For each outgoing link from node n, the repairer checks if the remaining bandwidth of the link, b_e, is sufficient (i.e. $b_e \geq B_{req}$) (line 9). Then, the destination n' is checked if it has been visited or belongs to blacklist $\mathbb{B}$ (line 11) since the Pareto-optimal path are simple paths as shown in Lemma 2 and there is no feasible path from nodes in the blacklist. The process presented from line 12 to line 15 is to verify if the current link could belong to a non-dominated path following Lemma 1. When the set of candidate links $\mathcal{E}^*$ is empty, the algorithm backs to the last selected vertex (line 16 and 17). Otherwise, repairing procedure selects a link arbitrarily among common entries of $\tilde{\mathcal{E}}$ and $\mathcal{E}^*$ to enhance randomness in searching solutions (line 20 to line 21).

Algorithm 1: Repairer

1 **Input:** An unfeasible path from S to D
2 **Output:** A feasible path from S to D
3 **foreach** *active bit of the input* **do** $\tilde{\mathcal{E}} \leftarrow e$;
4 **foreach** *QoS measure* i **do** Run Dijkstra for QoS measure $i \rightarrow c_i^{n,d}$;
5 $\mathcal{V}^* \leftarrow s$, $n \leftarrow s$, and $\mathbb{B} \leftarrow \emptyset$;
6 **while** $n \neq d$ **do**
7 $\mathcal{E}^* \leftarrow \emptyset$;
8 **foreach** e *originated from* n **do**
9 **if** $b_e < B_{req}$ **then continue** ;
10 $n' \leftarrow Dest(e)$;
11 **if** $n' \in \mathcal{V}^* \cup \mathbb{B}$ **then continue** ;
12 **foreach** *QoS measure* i **do**
13 $u_i^{n,d} \leftarrow \max_{j \neq i} c_{j,i}^{n,d}$;
14 **if** $c_{i,i}^{n',d} + c_i^{n,n'} \leq u_i^{n,d}$ **then**
15 $\mathcal{E}^* \leftarrow e$ and **break**;
16 **if** $\mathcal{E}^* = \emptyset$ **then**
17 $\mathbb{B} \leftarrow \mathbb{B} \cup \{n\}$, $n \leftarrow$ last entry in $\mathcal{V}^*$;
18 **if** $n \neq s$ **then continue**;
19 **else break**;;
20 **if** $\mathcal{E}^* \cap \tilde{\mathcal{E}} \neq \emptyset$ **then** Select e from $\mathcal{E}^* \cap \tilde{\mathcal{E}}$ randomly ;
21 **else** Select e among entries in $\mathcal{E}^*$ randomly ;
22 $n \leftarrow Dest(e)$ and $\mathcal{V}^* \leftarrow \mathcal{V}^* \cup \{n\}$;
23 **return** $\mathcal{V}^*$;

The worst case complexity of the code in line 11 is $\mathcal{O}(\log |\mathcal{V}|)$ and it is $\mathcal{O}(M)$ for the code from line 12 to line 15. Consequently, the overall worst case complexity of

the main loop of repairing procedure (line 6 to line 20) is $\mathcal{O}(|\mathcal{V}|^3 (M + \log|\mathcal{V}|))$. The complexity of Dijkstra algorithm (line 4) is $\mathcal{O}(M|\mathcal{E}|\log|\mathcal{V}|)$. Note that this loop can run once and reused for all individuals. Meanwhile, the main loop has to be executed for each individual.

C. Non-dominated crossover

Crossover is the process of combining the bits of one parent with those of another, in order to create an offspring that inherits characteristics of both parents. In this section, we present a proposed crossover scheme so as to create offspring that are not dominated by their parents.

We define a crossover point (CP) $n^{i,j}$ of a pair of paths (i, j) as a common vertex of both paths (excluding the source and the destination). A cross over point $n^{i,j}$ divides a path i to two sub-paths $P_{ia}\left(n^{i,j}\right)$, from source to $n^{i,j}$ (inclusive), and $P_{ib}\left(n^{i,j}\right)$, form $n^{i,j}$ (exclusive) to d. A non-isolated CP, $n^{i,j}$, is a CP of paths P_i and P_j such as $P_{ia}(n^{i,j}) \cap P_{jb}(n^{i,j}) \neq \emptyset$. Meanwhile, an isolated CP of a pair of parents, $m^{i,j}$, is a CP of paths P_i and P_j such as $P_{ia}(m^{i,j}) \cap P_{jb}(m^{i,j}) = \emptyset$. Note that a non-isolated (isolated) CP $n^{i,j}$ might be not a non-isolated (isolated) CP of (j, i) since it could happen that $P_{ja}(n^{i,j}) \cap P_{ib}(n^{i,j}) = \emptyset$ ($P_{ja}(n^{i,j}) \cap P_{ib}(n^{i,j}) \neq \emptyset$).

Lemma 3. *Given an offspring A formed by a non-isolated CP $n^{i,j}$, there exists an offspring formed by an isolated CP that dominates A.*

Proof. Since $n^{i,j}$ is a non-isolated CP, it means that $P_{ia}(n^{i,j}) \cap P_{jb}(n^{i,j}) \neq \emptyset$. Let us denote $\mathcal{C} = P_{ia}(n^{i,j}) \cap P_{jb}(n^{i,j})$ as the common vertices of $P_{ia}(n^{i,j})$ and $P_{jb}(n^{i,j})$. Obviously, it exists an isolated CP $m^{i,j} \in \mathcal{C}$, and forms two new sub-paths $P_{ia}(m^{i,j})$ and $P_{jb}(m^{i,j})$. Due to $P_{ia}(m^{i,j}) \cap P_{jb}(m^{i,j}) = \emptyset$ and additive QoS measures, $c^{P_{ia}(m^{i,j})} \leq c^{P_{ia}(n^{i,j})}$ and $c^{P_{jb}(m^{i,j})} \leq c^{P_{jb}(n^{i,j})}$. Consequently, the offspring formed by $(P_{ia}(m^{i,j}), P_{jb}(m^{i,j}))$ dominates the offspring $(P_{ia}(n^{i,j}), P_{jb}(n^{i,j}))$. □

Alg. 2 presents the proposed non-dominated crossover. The first step is to find common nodes from parents (line 3). Then, the crossover process checks each node in the set of common nodes to see if swapping at a common node can create the new offspring that are not dominated by their parents (line 6 to line 13). For each node in the set of common nodes, the crossover algorithm checks whether this node is an isolated CP (line 10 to line 11) since the non-isolated CPs are dominated as shown in Lemma 3. Then, each offspring is compared with two parents (line 12 to line 13). All isolated crossover that forms non-dominated offspring are stored in $\mathcal{S}_1$ and $\mathcal{S}_2$ corresponding to the first and second offspring. Finally, the algorithm returns random offspring among candidates (line 14 to line 15).

The computational complexity of finding common nodes in parents is $\mathcal{O}(|\mathcal{V}|\log|\mathcal{V}|)$. The worst case complexity of the process from line 10 to line 11 is $\mathcal{O}(|\mathcal{V}|\log|\mathcal{V}|)$. The comparison between offspring and parents has the complexity of $\mathcal{O}(M)$. Consequently, the loop from line 6 to line 13 has the complexity of $\mathcal{O}(|\mathcal{V}|(|\mathcal{V}|\log|\mathcal{V}| + M))$. The overall complexity of non-dominated crossover procedure is $\mathcal{O}(|\mathcal{V}|(|\mathcal{V}|\log|\mathcal{V}| + M))$.

Algorithm 2: Non-dominated Crossover

1 **Input:** Linked list of vertices of parents $\mathcal{P}_1, \mathcal{P}_2$
2 **Output:** non-dominated offspring
3 Find common vertices $\rightarrow \mathcal{V}_c$;
4 List of swap nodes $\mathcal{S}_1, \mathcal{S}_2$;
5 **if** $\mathcal{V}_c = \emptyset$ **then return** $\mathcal{P}_1, \mathcal{P}_2$;
6 **foreach** $n \in \mathcal{V}_c$ **do**
7 $\quad \mathcal{P}_i^{'}(n) \leftarrow \emptyset, i = 1, 2$
8 $\quad \mathcal{P}_{ia}(n) \leftarrow \mathcal{P}_i.sublist\,(s, n)\,, i = 1, 2;$
9 $\quad \mathcal{P}_{ib}(n) \leftarrow \mathcal{P}_i.sublist\,(n.next(), d)\,, i = 1, 2;$
10 $\quad$ **if** $\mathcal{P}_{1a}(n) \cap \mathcal{P}_{2b}(n) = \emptyset$ **then** $\mathcal{P}_1^{'}(n) = \mathcal{P}_{1a}(n) \cup \mathcal{P}_{2b}(n)$;
11 $\quad$ **if** $\mathcal{P}_{2a}(n) \cap \mathcal{P}_{1b}(n) = \emptyset$ **then** $\mathcal{P}_2^{'}(n) = \mathcal{P}_{2a}(n) \cup \mathcal{P}_{1b}(n)$;
12 $\quad$ **if** $\mathcal{P}_1^{'}(n) \neq \emptyset$ *and not dominated* **then** $\mathcal{S}_1 \leftarrow n$;
13 $\quad$ **if** $\mathcal{P}_2^{'}(n) \neq \emptyset$ *and not dominated* **then** $\mathcal{S}_2 \leftarrow n$;
14 Select node n_1^* in $\mathcal{S}_1$ randomly $\rightarrow$ OffSpring1 $= \mathcal{P}_1^{'}(n_1)$;
15 Select node n_2^* in $\mathcal{S}_2$ randomly $\rightarrow$ OffSpring2 $= \mathcal{P}_2^{'}(n_1)$;
16 **return** OffSpring1 and OffSpring2;

Parameters	Value
Population size	20
Number of generations	120
QoS metrisc	Loss rate and Latency
Loss rate	0.01% – 0.1% [19]
Number of runs per scenario	30
Confidence interval	95%
Simulation Environment	Intel Core i5-6300U, 8GB RAM
Genetic algorithm framework	jMetal [20]

TABLE I: Simulation parameters

VI. Numerical results

In this section, the performance of proposed schemes under different combinations of non-dominated crossover, single-point crossover, and mutation will be presented in order to determine appropriate configurations for different scenarios. In single-point crossover, both parents are swapped at a random CP. The new offspring is created by concatenating the first part of one parent with the second part of the other. NSGA-II is the GA algorithm as default. Simulation parameters are presented in Tab. I

We consider different scales of networks: from one to over seven hundreds of nodes. The sizes of networks and shortest distance in number of hops between source and destination are described in Table II. These topologies are referred to the dataset provided by topology-zoo [21] with real latency so as to keep the assessments practical. For GAs, we consider different combinations of proposed modules: Non-dominated crossover and mutation (NDC-M), Single-point crossover and mutation (SP-M), Non-dominated crossover only (NDC), and Mutation only (M). For NDC, it is unnecessary to execute repairing process after crossover since the offspring is a feasible solution, therefore reducing computational complexity.

Name	# nodes	# links	distance (# hops)
Kdl (Kentucky Data Link)	754	1788	39
Colt (Colt telecom)	153	354	18
GtsCe (GTS Central Europe)	149	386	27
UsCarrier (US Carrier)	158	378	26
Deltacom (ITC Deltacom)	113	322	18

TABLE II: Topology parameters

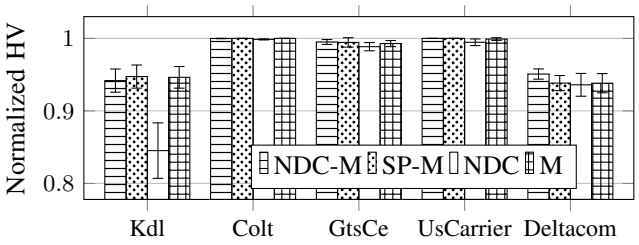

Fig. 3: Normalized Hypervolume with 2 QoS metrics

The conventional SAMCRA (SCR) defines the length as the maximum of ratios of metrics and their upper-bounds. Consequently, the output of SCR is not a Pareto-optimal solution. To derive Pareto-optimal solutions, we define the length as a vector of ratios instead of a linear or non-linear function. This modified SCR with Look-ahead feature, denoted as SAMCRA-LA-MO (SCRM), is used as reference since its output is the Pareto optimal solution. The hypervolume (HV) of a set of solutions S is the volume of objective space dominated by the set, limited by a given reference point. In other words, the greater HV means the better solutions. The normalized HV is defined as follows $\texttt{NHV} = \frac{\texttt{HV}_{GA}}{\texttt{HV}_{SCRM}}$, where $\texttt{HV}_{SCRM}$ is the HV of SCRM which is the optimal solutions and $\texttt{HV}_{GA}$ is the HV of GA-based solutions. The normalized HV (NHV) expresses how good the GA-based solutions are. We evaluates the performance of the proposed GA scheme by NHV, QoS-metrics, and calculation time.

Comparisons of four configurations in terms of NHV and calculation time are shown in Fig. 3 and Fig. 4. Even though the population size and the number of generations are small (20 and 120, respectively), the normalized HVs are over 80% in all cases. It means that the GA-based solutions are near to the optimal solutions. The performances of all configurations in most cases (except Kdl) are similar to each other. There are noticeable gaps between NDC and other configurations in Kdl scenario. It is because the missing of mutation and small population limits the capability of finding a Pareto optimal solution of NDC, especially in large network such as Kdl. However, the calculation time of NDC is shortest due to its simplicity. The gap in calculation time is proportional to the networks' complexity, since the repairing processes of high complex networks requires more calculation time. By deploying mutation, the combination NDC-M has the better performance. Nevertheless, the performance of NDC-M is similar to SP-M and M while it requires noticeable extra run-time. The latency and loss-rate of optimal solutions (Kdl-Pareto Front (Kdl-PF), Colt-Pareto Front (Colt-PF), GtsCe-Pareto Front (GtsCe-PF)) obtained by SCRM and GA-based solutions with NDC-M (Kdl-NDC-M, Colt-NDC-M, GtsCe-NDC-M) are shown in Fig. 5. The results are consistent with the normalized HV shown in Fig. 3. In Colt and GtsCe networks, the GA-based results are similar to optimal solutions. The solutions of Kdl-NDC-M is near to the optimal solutions; however, they do not cover all Pareto Front. That explains the lower normalized HV in Fig. 3.

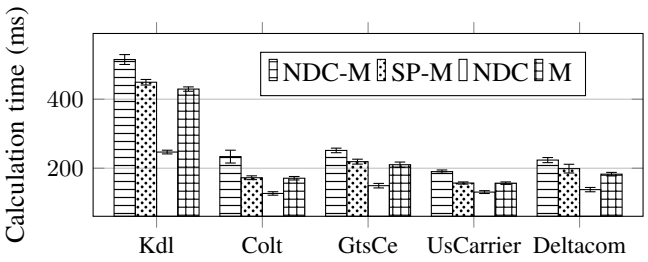

Fig. 4: Calculation time when there are 2 QoS metrics

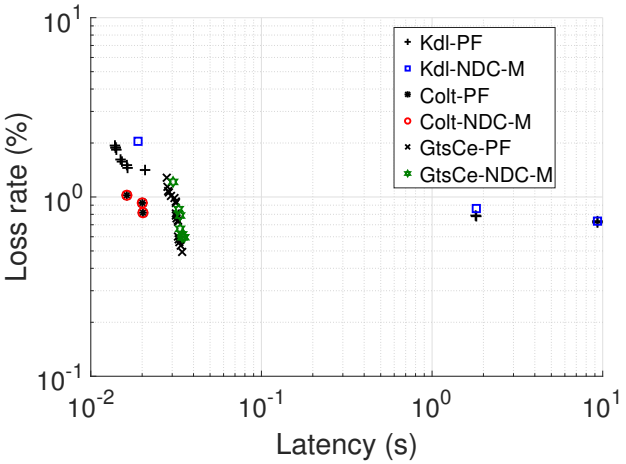

Fig. 5: Loss rate and Latency

Fig. 6 presents the calculation time of the proposed mechanism, and SCRM. The calculation time of SCRM is similar to SP-M when the sizes of networks is small or moderate. However, the calculation time of SCRM is much greater than the proposed mechanism in a large-scale network scenario, e.g. Kdl. As a result, the implementation of SCRM in large-scale networks could be impractical. The proposed mechanism can offer similar normalized HV as shown in Fig. 3 within 1s.

Above simulations confirm the performance of proposed

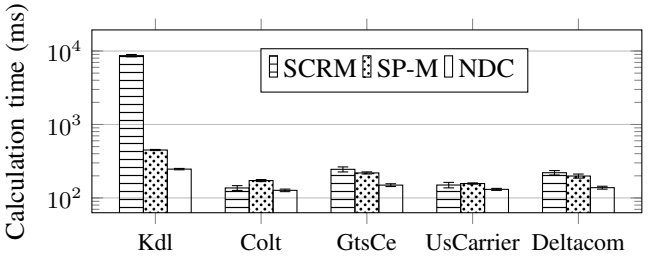

Fig. 6: Calculation Time of SCRM vs GAs with 2 QoS metrics

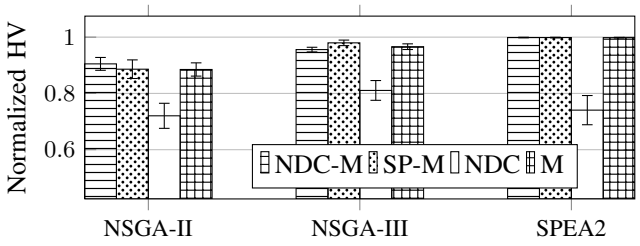

Fig. 7: Normalized HV (10 individuals and 60 generations), 2 QoS metrics of different GA algorithms

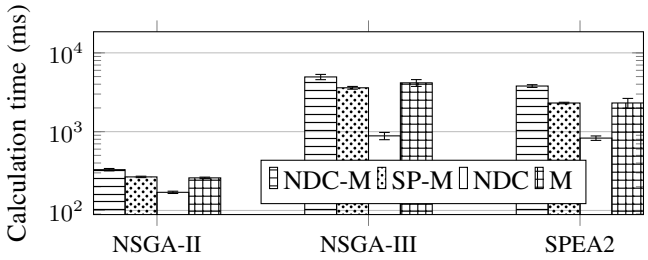

Fig. 8: Calculation time (10 individuals and 60 generations), 2 QoS metrics of different GA algorithms

mechanism with NSGA-II. Fig. 7 and Fig. 8 present the comparisons of the proposed mechanism with different GAs in Kdl scenario. NSGA-II has the worst performance in normalized HV but it is the best in the calculation time due to its simplicity. SPEA2 has better normalized HV and lower calculation time than NSGA-III. However, both SPEA2 and NSGA-III have the calculation time about 10 times that of NSGA-II.

VII. Conclusions

Multi-objective QoS routing is one of major challenges in networking. We proposed an efficient GA-based routing to cope with Multi-objective QoS routing by introducing the repairer concept and the non-dominated crossover operator. We confirmed the gain of the proposed mechanism through intensive simulations. A comprehensive study on how to control parameters so as to balance between calculation time and the quality of solution will be considered in future. Furthermore, we intend to extend this approach to a broader category of problems where path computation takes part, like VNF chains placement problem. For reducing computation time, multi-objective evolutionary algorithms that enable fully parallelization will be considered in future.

References

[1] F. A. Kuipers and P. Van Mieghem, "Bi-directional Search in QoS Routing," in *Proceedings of Quality for All: QoFIS 2003*, Stockholm, Sweden, Oct. 2003, pp. 102–111.

[2] F. Kuipers, P. V. Mieghem, T. Korkmaz, and M. Krunz, "An overview of constraint-based path selection algorithms for QoS routing," *IEEE Communications Magazine*, vol. 40, no. 12, pp. 50–55, Dec 2002.

[3] B. Lorenzo and S. Glisic, "Optimal routing and traffic scheduling for multihop cellular networks using genetic algorithm," *IEEE Transactions on Mobile Computing*, vol. 12, no. 11, pp. 2274–2288, Nov 2013.

[4] J. W. Guck, A. V. Bemten, M. Reisslein, and W. Kellerer, "Unicast QoS Routing Algorithms for SDN: A Comprehensive Survey and Performance Evaluation," *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2017.

[5] H. D. Neve and P. V. Mieghem, "TAMCRA: a tunable accuracy multiple constraints routing algorithm," *Computer Communications*, vol. 23, no. 7, pp. 667 – 679, 2000.

[6] P. V. Mieghem, H. D. Neve, and F. Kuipers, "Hop-by-hop quality of service routing," *Computer Networks*, vol. 37, no. 3, pp. 407 – 423, 2001.

[7] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," in *IEEE CEC*, Jun 1994, pp. 82–87 vol.1.

[8] N. Srinivasan and K. Deb, "Multi-objective function optimisation using non-dominated sorting genetic algorithm," *Evolutionary Comp*, vol. 2, no. 3, pp. 221–248, 1994.

[9] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr 2002.

[11] Z. He and G. G. Yen, "Many-objective evolutionary algorithm: Objective space reduction and diversity improvement," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 145–160, Feb 2016.

[12] K. Deb and H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, Aug 2014.

[13] A. Farrel, J.-P. Vasseur, and J. Ash, "A path computation element (PCE)-based architecture," Tech. Rep., 2006.

[14] J. L. Le Roux, "Path computation element (PCE) communication protocol (PCEP)," 2009.

[15] Netphony pce repository v1.3.3. [Online]. Available: https://github.com/telefonicaid/netphony-pce

[16] P. V. Mieghem and F. A. Kuipers, "Concepts of exact qos routing algorithms," *IEEE/ACM Transactions on Networking*, vol. 12, no. 5, pp. 851–864, Oct 2004.

[17] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, Sep 1996.

[18] Gurobi. [Online]. Available: http://www.gurobi.com/

[19] E. Rodriguez-Colina, D. Gil-Leyva, J. L. Marzo, and V. M. RamosR., "A bit error rate analysis for tcp traffic over parallel free space photonics," *Telecommunication Systems*, vol. 56, no. 4, pp. 455–466, Aug 2014. [Online]. Available: https://doi.org/10.1007/s11235-013-9764-4

[20] jMetal 5. [Online]. Available: https://jmetal.github.io/jMetal/

[21] The internet topology zoo. [Online]. Available: http://www.topology-zoo.org/dataset.html