A Two-level Genetic Algorithm for Inter-domain Path Computation under Node-defined Domain Uniqueness Constraints

Do Tuan Anh
School of Information and Communication Technology
Hanoi University of Science and Technology
Hanoi, Vietnam
anhdt@soict.hust.edu.vn

Huynh Thi Thanh Binh
School of Information and Communication Technology
Hanoi University of Science and Technology
Hanoi, Vietnam
binhht@soict.hust.edu.vn

Nguyen Hoang Long
School of Information and Communication Technology
Hanoi University of Science and Technology
Hanoi, Vietnam
longnh.mso@gmail.com

Ta Bao Thang
School of Information and Communication Technology
Hanoi University of Science and Technology
Hanoi, Vietnam
tabaothang97@gmail.com

Su Simon
DEVCOM Army Research Laboratory
United States of America
simon.m.su.civ@mail.mil

***Abstract*—Recent years have witnessed an increment in the number of network components communicating through many network scenarios such as multi-layer and multi-domain, and it may result in a negative impact on resource utilization. An urgent requirement arises for routing the packets most efficiently and economically in large multi-domain networks. In tackling this complicated area, we consider the Inter-Domain Path Computation problem under Node-defined Domain Uniqueness Constraint (IDPC-NDU), which intends to find the minimum routing cost path between two nodes that traverses every domain at most once. Owing to the NP-Hard property of the IDPC-NDU, applying metaheuristic algorithms to solve this problem usually proves more efficient. In like manner, this paper proposes a Two-level Genetic Algorithm (PGA), where the first level determines the order of the visited domains, and the second level finds the shortest path between the two given nodes. Furthermore, to facilitate the finding process, a method to minimize the search space and a new chromosome encoding that would reduce the chromosome length to the number of domains are integrated into this proposed algorithm. To evaluate the efficiency of the proposal, experiments on various instances were conducted. The results demonstrated that PGA outperforms other algorithms and gives results no more than twice the optimal values.**

***Index Terms*—Multi-Domain, Path Computation Element, Evolutionary Algorithm.**

I. Introduction

In this day and age, computer networks are growing progressively more sophisticated, and therefore, they are often partitioned into smaller domains for effortless management. Along with this comes the path computation of packets between domains, which is a problem attracting many not only the research communities but also the government and people in other spheres. In multi-domain networks, special computational components and cooperation between elements in different domains may be required to do knotty path computation. First introduced in RFC 4655 [1], Path Computation Element (PCE) architecture has been promoted as a dedicated network entity to overcome this problem. On behalf of network nodes, the PCE gathers link-state information and performs path computation. Nevertheless, the path computational responsibility of each PCE is restricted to its domain. Since individual PCE is unable to identify all the links and resources in every domain, calculating an end-to-end traffic engineered path is a challenging hurdle in a multi-domain network environment. Several solutions have been proposed to overcome the drawback of the PCE architecture, for instance, Backward Recursive PCE-Based Computation (BRPC) protocol. Although this protocol helps PCEs with exchanging data among their neighbors, it reveals another flaw that the order of the crossed domains must be pre-calculated, making it more arduous to achieve the computational path's optimality. Therefore, Hierarchical Path Computation Element (h-PCE) architecture has been employed to concurrently combine the computation of end-to-end paths and the optimization of passed domains sequence. According to RFC 6805 [2], in the h-PCE architecture, a parent PCE manages a domain topology map that contains the child domains and their interconnections. For avoiding transgressing the confidentiality condition, the parent PCE is not allowed to

978-1-7281-8393-0/21/$31.00

know about the resource availability within the child domains and the connectivity availability across each domain. However, h-PCE is only applicable to circumstances with minute groups of domains. Paolucci et al. [3] surveyed the PCE architectures and their operation in path computation for more information.

In a recent research article on multi-domain, L.Maggie et al. [4] improved h-PCE by introducing a new problem of Inter-Domain Path Computation problem under Domain Uniqueness constraint (IDPC-DU), which concentrates on detecting a minimum cost path that traverses every domain at most once. Two different variations of the problem are put forward for domain definitions, namely IDPC-EDU and IDPC-NDU. More specifically, for the IDPC-EDU, domains are assigned on edges, while for the IDPC-NDU, they are distinguished on nodes. Overall, the Domain Uniqueness Constraint (DU) constraint boosts the packet processing ability and reduces delays and signaling overhead, which is entirely consistent with the h-PCE architecture in practice. Appertaining to the NP-Completeness, the IDPC-DU's real computational bottleneck, however, lies behind the number of domains [4]. Commencing with the idea of reducing this problem's complexity, the authors proposed a clustering concept that artificially decrements the number of domains without loss of optimality. Moreover, a dynamic programming approach whose complexity is $O(|V|^2 2^{|D|} |D|^2)$ (where V is the number of nodes, and D is the number of domains), is presented to solve the IDPC-DU along the lines of the classic Bellman-Ford algorithm.

Nonetheless, owing to the IDPC-DU's NP-Hard property, all manner of approaches can be applied to tackle this problem more efficiently, especially using metaheuristic algorithms [5, 6]. Belonging to this class, Evolutionary Algorithms (EAs) have substantiated their effectiveness in addressing NP-Hard problems. EAs stem from the notion that natural evolution is the most perfect, logical and optimal process in and of itself. One of the most representative of EAs is Genetic Algorithm (GA). The algorithm commences with a population of individuals that undergo crossover and mutation to produce offspring. The procedure iterates itself intending to preserve genetic material, making an individual fitter concerning a given environment while eliminating weaker ones. The optimization of evolution progress guarantees that the next generation is always better than the previous one. One advantage of using GA is that it maintains and handles a set of solutions that we usually call a population, while other searching methods manipulate only one point in the search space.

To the best of our knowledge, a solution to the IDPC-EDU has been suggested, whereas there has not been any endeavor to ascertain solutions to the IDPC-NDU. For this reason, this paper introduces a new proposal to solve this problem. Generally, one of the intricacies related to the IDPC-DU is that we have to detect the shortest path in a massive search space. To handle this task, we propose a new approach that remodels the IDPC-NDU into two sub-problems, one of which is solved by a metaheuristic algorithm called GA, and the other is addressed by an exact algorithm. As a result, the search space would be diminished in dimensionality compared to the original problem, hence lessening the consumed resource and computational time.

The major contributions of this work can be summarized as follows:

- Propose a pre-filtering process for reducing the search space of the IDPC-NDU.
- Propose an approach to solve the IDPC-NDU by decoupling it into two sub-problems.
- Combine two algorithms to tackle the IDPC-NDU. In simple terms, we propose GA to solve the first sub-problem of the IDPC-NDU and suggest using Dijkstra's algorithm to overcome the second one.
- Devise a method to represent a chromosome, which reduces the chromosome length to the number of domains, and a corresponding decoding mechanism.
- Analyze and estimate experimental results on various test instances to portray the efficiency of the suggested model.

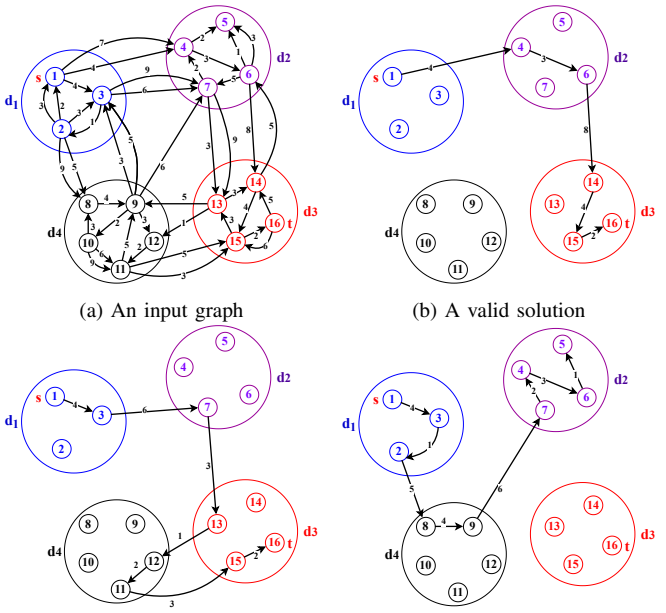
The rest of this paper is organized as follows. Section II describes some definitions and notations of the problem. Section III presents the related works, while section IV elaborates on the proposed algorithms. Section V includes the setups of our experiments, computational results on various test sets, and a comparison of performance with other algorithms. Conclusions and discussions on future extension are given in Section VI.

II. Notation and definitions

The IDPC-NDU's network model is considered a weighted directed multigraph $G(V, E, w, s, t)$ which permits to have multiple edges between two vertices. Set of vertices V is partitioned into K separate clusters interpreted as domains on the network. The objective of the IDPC-NDU is to find a minimum cost path from source vertex s to target vertex t such that each domain is visited at most once. The IDPC-NDU is stated as follows:

Input:	- A weighted directed multi-graph $G = (V, E, w)$, - Vertex set V is divided into K domains $\{V_1, V_2, ..., V_K\}$, $V_i \cap V_j = \emptyset$. - A source vertex s. - A target vertex t.
Output:	A path $(p_0, p_1, \ldots, p_h)$ where $p_0 = s$ and $p_h = t$, in G.
Constraint:	Node-defined Domain Uniqueness (NDU): once p has left a domain, it does not revisit it later on. If $p_i \in V_d$ and $p_{i+1} \notin V_d$, then $p_{i+k} \notin V_d$ $\forall k > 1, 1 \leq d \leq K, p_i \in V$.
Objective:	$\sum_{i=0}^{h-1} w(p_i, p_{i+1}) \rightarrow$ minimum.

An example of the IDPC-NDU is illustrated in Figure 1. Figure 1(a) describes an input graph G with 16 vertices partitioned into 4 domains. The IDPC-NDU finds the optimal path from source vertex 1 to target vertex 16 that obeys the



(a) An input graph

(b) A valid solution

(c) An invalid solution due to violating NDU constraint

(d) An invalid solution that is unable to reach the destination node

Fig. 1: An example of input graph and solutions of IDPC-NDU

NDU constraint. A feasible path of the IDPC-NDU is depicted in Figure 1(b). Note that it does not need to visit all domains. Figure 1(c) and Figure 1(d) show two invalid solutions of the IDPC-NDU:

- A path does not satisfy the NDU constraint because there exists a domain visited more than once such as domain 4 in Figure 1(c).
- A path does not terminate at the destination node, leading to an invalid solution such as in Figure 1(d).

III. Related Works

Shortest Path Problem (SPP) so far has always been well-known with a multitude of diverse variations and solutions introduced [7]. For example, in [8], Horváth et al. proposed several new components to deal with Resource Constrained Shortest Path Problem (RCSPP), of which variable-fixing and heuristic solution search are the two most important factors. Each operator had been experimented in both separate and combination. Luigi Di et al. [9] addressed the elementary Shortest Path Problem with Forbidden Paths (SPPFP), which is also another instance of the SPP, by defining two different approaches, including a Branch & Bound-based algorithm and a dynamic programming approach. Numerical results highlighted that the performance of the two strategies relies on additional constraints and the search space dimension.

Recently, metaheuristic algorithms have emerged as new approaches to solve this problem, for instance, Ant Colony Optimization (ACO) and Particles Warm Optimization Algorithm (PSO). In [10], Suholt et al. analyzed the ACO for Stochastic Shortest Path Problems (SSPP) with various setting of noisy edge. In the same year, the authors also studied the running time of ACO for SPP in directed graphs [11]. The authors in [12] investigated PSO in solving SPP by developing an original encoding and joining it with a heuristic operator. Nevertheless, the limitation of these approaches is that it is not suitable for multigraph problems or the ones with a large number of domains.

Therefore, some proposals hitherto selected to combine metaheuristic algorithms with exact algorithms, typically the Dijkstra's algorithm, and they have proven their effectiveness. One merit of Dijkstra's algorithm is its ability to scan the entire search space, thereby giving the best solution. One of the latest research that utilized the same method is Clustered Shortest-Path Tree Problem (CluSPT) [13, 14]. In these works, CluSPT is divided into two sub-problems where the first sub-problem is to locate an edge set that connects among the clusters, while the second one is to find a spanning tree for the sub-graph in each cluster. Specifically, the authors use GA and Multifactorial Evolutionary Algorithm (MFEA) to determine the edge set and work with Dijkstra's algorithm to find the best spanning tree. In like fashion, to remedy the IDPC-NDU, our suggestion is to decompose it into two slighter sub-problems mentioned above.

Latterly, MFEA has been employed to tackle the Inter-Domain Path Computation problem under Edge-defined Domain Uniqueness Constraint (IDPC-EDU) [15]. With its multitasking capability, MFEA is considered one of the optimal variations of the EAs. The varying number of edges among proximity nodes induces difficulty in implementing EAs to path computation, which is how to encode a chromosome precisely. In [15], the author used two-layer encryption, where one layer describes each node's priority, while the other depicts the index of edge that the path will take to reach another node. This encoding has the convenience of finding a valid solution without heeding whether an edge exists between two specific nodes, and it further facilitates exploiting the search space completely. Even though experimental results have shown the effectiveness of MFEA, this approach still presents deficiencies. To be more precise, this encoding method reveals a weakness that the length of the individual chromosome is equal to the number of nodes, which leads to a very complicated decoding mechanism.

Consequently, this paper offers a new algorithm based on GA to solve the IDPC-NDU. The proposal improves the quality of solutions compared to traditional algorithm.

IV. Proposed Algorithm

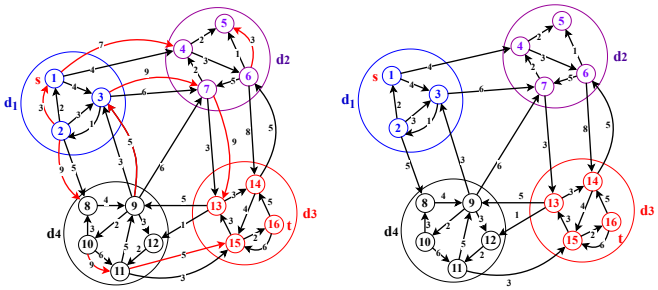
In this section, we present our approach based on GA termed PGA to tackle the IDPC-NDU in further detail. As mentioned above, we divide the IDPC-NDU into two sub-problems. The first sub-problem goal is to establish the order of domains that the path will go through, thereby abridging the original graph to a smaller one. Then, for each solution of the first sub-problem, we address the second one to locate the shortest path between two given nodes by an exact algorithm. The structure of PGA is presented in Algorithm 1.

It is clear that the objective function of the IDPC-NDU is to find the min-cost path. However, accomplishing this task

Algorithm 1: PGA algorithm for IDPC-NDU

```
1  begin
2      t ← 0;
3      Perform pre-filtering process to reduce search
         space of the problem;
4      P(0) ← Construct graph G_D and generate N
         random individuals  ▷ Refer to Subsection IV-B;
5      best ← the fittest individual in P(0);
6      while stopping conditions are not satisfied do
7          Offspring population P_c(t) ← ∅;
8          while |P_c(t)| < N do
9              a ← Randomize in range of (1, N/2);
10             b ← Randomize in range of (1, N);
               /* Crossover with probability
                  pc and Mutate with
                  probability pm                  */
11             if rand(0, 1) < pc then
12                 o_1, o_2 ← crossover(p_a, p_b)   ▷ Refer to
                     Section IV-D;
13                 for i ← 1 to 2 do
14                     if rand(0, 1) < pm then
15                         o_i ← mutate(o_i)        ▷ Refer to
                             subsection IV-E;
16             P_c(t) ← P_c(t) ∪ {o_1, o_2};
17         Construct and evaluate solution for each
             offspring in P_c(t) ▷ Refer to subsection IV-C;
           /* Survival Strategy                   */
18         P(t + 1) ← P_c(t);
19         Sort P(t+1) in descending order of fitness;
20         Replace a random individual in P(t+1) with
             best to keep elitist and update best;
21         t ← t + 1;
```

in a space with bazillion edges is a formidable obstacle and virtually unattainable due to the limitation of resources and time. Therefore, the pre-filtering process is devised to meet the need for minimizing the search space, whose intention is to convert the input multigraph to a simple one. This process is detailed as follows. If there exists more than one edge going from vertex i to vertex j, we retain only the one with the lowest weight and eliminate the remaining other. Besides, apart from domains containing the source and the destination node, any domains whose indegree or outdegree is equal to 0 will also be removed. The withdrawal of redundant edges and domains facilitates the searching process on the simple graph, hence improving the running time and reducing resources consumed. Figure 2(a) illustrates an example of an unfiltered weighted directed inter-domain multigraph $G = (V, E)$. As can be seen, between two vertices 1 and 4, there are two edges whose weights are 7 and 4, respectively. In this case, we keep only the second one, which has a lower cost. In the same way, we filter all unnecessary edges coloured in red, as shown in Figure 2(a). Figure 2(b) demonstrates a simple graph that results from the pre-filtering process.



(a) Select red edges to be filtered (b) Simple graph after filtered

Fig. 2: Pre-filtering process

The implementation steps of the algorithm are discussed in detail in the following subsections.

A. Individual Representation

As can be seen in Figure 2(b), if a path satisfies the NDU constraint, that path visits vertices, which lie in the same domain as the source node, then move to another domain and reiterates this process until it reaches the destination node. In other words, for every domain, there is at most one path entering and leaving it. In agreement with this, this paper innovates an encoding method to represent a chromosome as an order of visited domains. Thus, the length of each chromosome is reduced to the number of domains, which is drastically smaller than the number of nodes.

Initially, we construct a simple directed graph $G_D = (V_D, E_D)$ where $V_D = \{d_1, d_2, ..., d_K\}$ is the set of domains in graph G, and $E_D = \{e_{ij}\}$ where e_{ij} is an edge connecting two vertices d_i and d_j, only if there is a directed edge that links two domains d_i and d_j respectively in graph G. Figure 3 delineates graph G_D, which constructs from G.

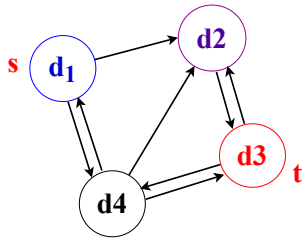


Fig. 3: Simple graph G_D

Secondly, we determine the domain visit order, starting from the domain containing the source node and travelling to the domain containing the destination node. To ensure the length of all chromosomes are fixed as well as avoid the circumstance of non-existence connection between two domains, we utilize Priority-based Encoding (PE). In this encoding, an array of integers, whose each element specifies a domain's priority, represents a chromosome. From a domain, we only consider the priority values of adjacent ones, and a higher-priority domain is visited earlier than others. The application of PE in this work is illustrated in Figure 4. In this example, the priority values from domain d_1 to domain d_4 are 2, 1, 3, and 4 in turns. We start from the domain containing the source

node, which in this case, is d_1. In graph G_D, both domains d_2 and d_4 connect to d_1, but the priority of domain d_2 is higher than that of d_4. Therefore, d_2 is chosen to be visited and, similarly, d_3 and d_4.

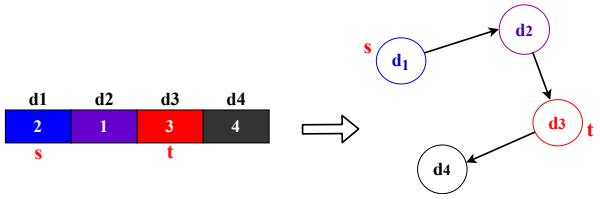


Fig. 4: A representation of a chromosome in graph

B. Individual Initialization Method

To gear towards optimizing the scannability in search space and maintaining the diversity of the population in GA, the priority of all crossed domains in each individual are generated randomly. We denote the representation of a chromosome as $I = (D_1, D_2, ..., D_K)$ where $\{D_1, D_2, ..., D_K\}$ is a permutation of $(1, 2, ..., K)$.

C. Decoding Method

In this section, corresponding to each chromosome, we use the Dijkstra's algorithm to find the shortest path from the source node to the destination node. However, the performance of this algorithm is not outstanding in graphs with a large number of edges. Therefore, in place of applying the Dijkstra's algorithm directly on graph G, we formulate to construct graph G' as follow: $G' = (V', E')$, where V' is the set of nodes from graph G, and E' is the E of G but removing all edges from domain d_i to d_j that do not satisfy the order of visited domains in the considered chromosome.

Take Figure 4 as an sample chromosome. According to this, the priority order of the domains d_1, d_2, d_3, and d_4 being $2, 1, 3$, and 4 in turns. It means the order of domains traversed is d_1, d_2, d_3, and d_4 correspondingly. Presuming a path p traveling through domains is in this order. Apparently, p contains an edge from d_1 to d_2, one from d_2 to d_3 and d_3 to d_4 analogously. On the other hand, p does not have any edge from d_3 to d_2. This assumption motivates us to build G'. First, we preserve all edges created by vertices in the same domain

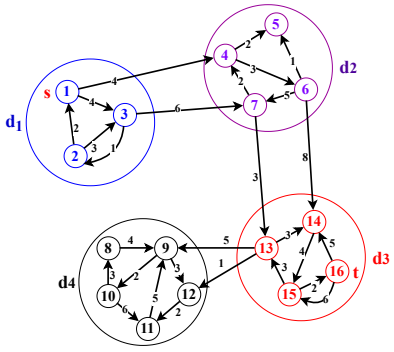


Fig. 5: An example of graph G' which is constructed from the chromosome in figure 4 and graph G

in G, such as $(1, 3)$ in d_1 and $(4, 5)$ in d_2. Subsequently, for inter-domain edges, we retain only edges from domain d_i to d_j based on the order of visited domains, specifically $(1, 4)$, $(3, 7)$ and $(7, 13)$. Figure 5 shows the graph G' which is constructed from the chromosome in Figure 4 and the initial graph G in Figure 2(b).

The rest is to locate the shortest path from the source node s to the destination node t while satisfying the NDU constraint. To handle this work, we apply Dijkstra's algorithm on G'. The shortest path's cost is later updated as the equivalent individual's fitness in the first sub-problem. Algorithm 2 demonstrates our decoding method in detail.

Algorithm 2: Proposed Decoding Method

Input:
- An input graph $G(V, E, s, t)$;
- An individual $I = \{D_1, D_2, \ldots, D_K\}$;

Output: A path $p = \{s, p_1, \ldots, p_h, t\}$

begin
  /* Construct a graph G' (V', E') */
  $V' \leftarrow V$;
  $d_s \leftarrow$ Determine a domain which contains source vertex s;
  $d_t \leftarrow$ Determine a domain which contains target vertex t;
  $E' \leftarrow$ All edges in the domain d_s;
  $cur \leftarrow d_s$;
  while $cur \neq d_t$ **do**
    $U \leftarrow$ Find a set of unvisited domains adjacent from cur in G;
    if U *is* $\emptyset$ **then**
      Return null;
    else
      $d_m \leftarrow$ the highest priority domain in U;
      $E' \leftarrow E' \cup \{$ a set of edges of the domain $d_m\}$;
      $E' \leftarrow E' \cup \{$ all edges directing from domain cur to $d_m\}$;
      $cur \leftarrow d_m$;
  /* Find the optimal path in G' */
  $p \leftarrow$ Apply Dijkstra's Algorithm to find the shortest path from s to t in G';
  Return p;

D. Crossover Operator

One problem encountered in solving the IDPC-NDU problem is ensuring the path found must fulfil the NDU constraint. To ensure compliance with this, in this section, we apply Order Crossover (OX). OX has been used multiple times in problems that applied GA and likewise have presented its productiveness. It constructs an offspring by choosing a substring of one parent and maintaining the other parent's elements' relative order. Details of the OX are described in an example shown in Figure 6.

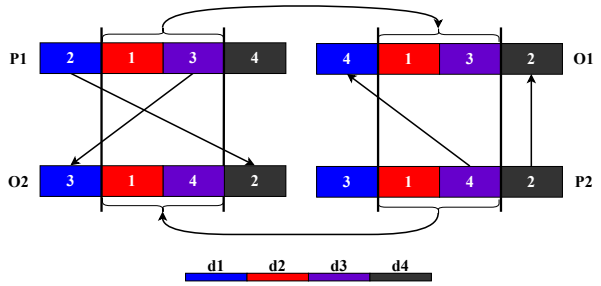


Fig. 6: An example illustrates how OX generates two offspring

In this example, we consider the two parents: P1 $(2\ 1\ 3\ 4)$ and P2 $(3\ 1\ 4\ 2)$. Two crossover points are first selected randomly to define the mappings section, which gives $(2|13|4)$ and $(3\,|\,1\,4\,|\,2)$. This section is position-intact copied from the parents to the corresponding offspring, hence $(*\,|\,1\,3\,|\,*)$ and $(*\,|\,1\,4\,|\,*)$. Next, commencing from the second cut point of the other parent, the rest of the elements are copied to the offspring in the order in which they appear but omitting the presented one. When reaching the end of the parent string, we continue from its first position. This example reproduces the following children: O1 $(4\,|\,1\,3\,|\,2)$ and O2 $(3\,|\,1\,4\,|\,2)$.

E. Mutation Operator

In contrast to crossover, observing the NDU constraint on a chromosome can be accomplished by using the Swapping Mutation (SW), which readily swaps the priority values of two domains. Figure 7 shows how the SW works on a representation. This example swaps the first and fourth domains' priority values, which results in an offspring: $(4\,|\,2\,3\,|\,1)$.

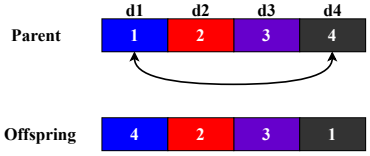


Fig. 7: An example illustrates how SW works

V. Computational Results

A. Problem Instances

On account of no public dataset to be available for the IDPC-NDU, two distinct types of instances are created based on the dataset of IDPC-EDU [15], which is also a shortest-path problem. We first generated three parameters for each instance: number of nodes, number of domains, and number of edges. After that, an optimal path p where the weight of edges is equal to 1 and the number of domains on p is approximately the input graph's domain number. Next, the noise is added to the instance by for every node in p, besides random weight edges, several random one-weight edges from that node to some other nodes not in p and some random edges with greater values of weight than the total cost of p are added into. These traps make simple greedy algorithms harder to find the optimal solution. Especially in Type 2, feasible paths whose length is less than three are removed. The datasets are categorized into two kinds regarding dimensionality: small instances, each of which has between 50 and 2000 vertices, and large instances, each of which has over 2000 vertices.

B. Experimental Setup

To evaluate the performance of the proposed algorithms PGA, this paper adopted an ACO algorithm named S-ACO [10] which has proven effective for the direct shortest path. For maximizing the performance of S-ACO, we use the same parameters to set up a similar experimental environment as the previous work. The algorithms were simulated 30 times on the computer (Intel Core i7 - 4790 - 3.60GHz, 16GB RAM) with a population size of 100 individuals evolving through 500 generations which means the total numbers of task evaluations are 50000, the random mating probability (pc) is 0.8 and the mutation rate (pm) is 0.05. The source codes were installed by Java language.

Let A be the set of tested algorithms, and I be the set of instances. Assume that S^i_{ar} be the cost value obtained by algorithm a in A, for instance i in I, on the r-th run. Let also B^i be the best-known solution value for instance i, among all algorithms and previous results. Then, we define Relative Percentage Differences (RPD) (for minimization problem) as

$$RDP^i_{ar} = \frac{S^i_{ar} - B^i}{B^i} * 100 \tag{1}$$

These RDPs are used to compute the statics such as box plots, medians, means and hypothesis tests for experiments below. The smaller RDP values are, the better results found by the algorithm.

C. Experimental Results

The performance of PGA for the IDPC-NDU is evaluated under two analyses:

- Analyse computational complexity of the algorithm.
- Analyse quality of the proposed algorithm in comparison with other algorithms and optimal solution.

1) Analyse computational complexity of the algorithm: Let MAX_GEN be the maximum number of iterations of the algorithm. In each iteration, the number of executions of the crossover, mutation, and decoding operators are $N, p_m * N$, and N times, respectively. The complexity of the algorithm is as follows:

$O(\text{PGA}) = O(\text{Pre-filtering process}) + O(\text{Initialization}) + MAX\_GEN \times \Big(N \times O(\text{Crossover}) + p_m \times N \times O(\text{Mutation}) + N \times O(\text{Decoding}) + O(\text{Selection}) \Big)$ in which the complexity of each component is as follows:

- In the pre-filtering process, each edge in graph is traversed one times. Therefore, the complexity of the pre-filtering process is $O(|E|)$ with $|E|$ is the number of edges in the initial graph.
- The initialization method needs a computational cost $O(KN)$ to generate N random individuals with K is the number of domains.

- The complexity of mutation, crossover and survival selection operator are $O(1)$, $O(K)$ and $O(NlogN)$ respectively.
- The decoding method needs to traverse all edges in each domain and edges connecting two consecutive domains to construct the graph G'. This causes a computational cost of $O(|E|)$. Then, the Dijkstra's algorithm applied on G' to find the shortest path has a running time complexity to $O(|E|+|V|log|V|)$. Therefore, the decoding method's complexity is $O(|E|+|V|log|V|)$.

Therefore, O(PGA) =

$$O(|E|)+O(KN)+MAX\_GEN\times\Big(N\times O(K)+p_m\times N\times O(1)+N\times O(|E|+|V|log|V|)+O(NlogN)\Big)$$

$$=O\Big(MAX\_GEN\times N\times(K+|E|+|V|log|V|+logN)\Big).$$

2) Analyse quality of the proposed algorithm in comparison with other algorithms and optimal solution: The experimental results shown in Table II and Table III have pointed out that the proposed PGA outperforms S-ACO on both the best value, average and standard deviation in all instances. The average improvement percentage of PGA in comparison to S-ACO as shown in Table I is 32% in Type 1 Small, 27% in Type 1 Large, 68.1% in Type 2 Small, and 68.9% in Type 2 Large. Overall, the proposed PGA improves S-ACO 49.8% in term of the average result and the biggest gap recorded is 83.3%.

TABLE I: The improvement percentage (PI) of PGA in comparison to S-ACO

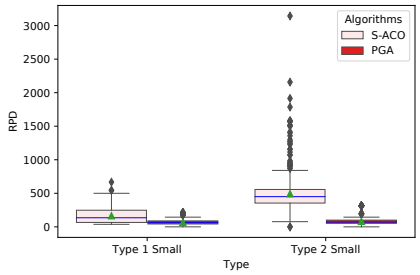
Type	Minimum PI	Average PI	Maximum PI
Tpye 1 Small	0.8	32	61.9
Type 1 Large	8.5	27	58.5
Tpye 2 Small	47	68.1	83.3
Type 2 Large	59.5	68.9	78.9
Entire instances	0.8	49.8	83.3

Furthermore, experimental results are presented in the form of box plots in terms of the RPD values in Figure 8, in which the first quartile (LQ) and third quartile (UQ) are the bottom and top of the box, respectively, and the line inside the box denotes the median. The inter-quartile range (IQR) is the difference between UQ and LQ. The smaller RPD values are, the better performance the algorithms draw. All statistics (LQ, UQ, median) of PGA are lower than S-ACO. This shows that the results of the PGA are more stable and accurate than S-ACO in both small and large instances.

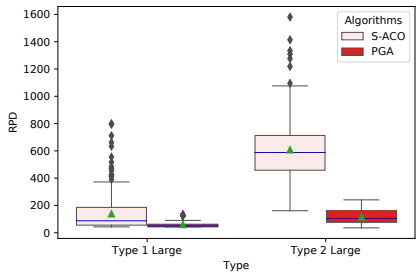
Finally, Wilcoxon signed-rank tests are conducted based on the obtained RPD values as shown in Table IV to find a factor that the obtained solution is not far from the optima. All obtained p-values are less than $\alpha = 0.05$, which confirms a significant difference between pairs of algorithms. The positive decision (+) means that PGA outperforms another algorithm with statistical significance. Besides, the negative decision (-) means that the non-existence of evidence ensures the statistical significance of the performance gap. The obtained statistical values indicate that the results of PGA are at most twice the optimal results.

TABLE II: The obtained results of all algorithms in Type 1

	Instances	S-ACO			PGA			OPT
		BF	AVG	STD	BF	AVG	STD	
Type 1 Small	idpc_ndu_1002_12_82252	14.0	14.4	0.56	8	13.80	1.10	8
	idpc_ndu_1002_22_36564	37.0	42.6	1.96	30	34.00	2.86	18
	idpc_ndu_1192_19_37744	54.0	70.8	4.57	40	47.30	5.06	18
	idpc_ndu_1202_22_65521	22.0	25.4	9.47	22	22.00	0.00	16
	idpc_ndu_1256_21_44446	73.0	96.4	10.82	46	47.97	2.74	26
	idpc_ndu_1322_22_48821	44.0	84.6	33.77	44	44.00	0.00	26
	idpc_ndu_1506_9_130556	19.0	23.7	5.69	11	12.33	3.03	11
	idpc_ndu_1514_16_78292	33.0	68.7	9.29	33	33.00	0.00	21
	idpc_ndu_1514_30_78351	30.0	37.3	18.98	30	30.00	0.00	21
	idpc_ndu_1602_22_60574	65.0	122.9	25.79	46	46.80	0.41	24
	idpc_ndu_1682_23_67612	76.0	80.2	7.07	46	55.03	11.45	24
	idpc_ndu_1730_20_88509	54.0	62.2	14.90	39	41.80	0.76	24
	idpc_ndu_2002_22_178026	24.0	24.2	0.76	24	24.00	0.00	16
	idpc_ndu_427_7_14927	13.0	15.9	3.59	8	8.00	0.00	8
	idpc_ndu_704_15_16990	32.0	57.5	25.32	31	31.00	0.00	21
	idpc_ndu_842_23_31617	22.0	22.2	0.76	22	22.00	0.00	16
Type 1 Large	idpc_ndu_2102_23_164811	27.0	29.5	12.00	27	27.00	0.00	18
	idpc_ndu_2402_22_260967	24.0	27.9	6.55	24	24.00	0.00	16
	idpc_ndu_2494_12_276620	23.0	31.3	6.96	23	23.00	0.00	16
	idpc_ndu_2502_22_229601	29.0	37.1	4.68	29	29.00	0.00	18
	idpc_ndu_2522_20_171940	63.0	98.8	24.88	41	41.00	0.00	18
	idpc_ndu_2715_22_592246	13.0	14.7	1.81	13	13.00	0.00	8
	idpc_ndu_2918_29_293109	30.0	42.8	14.47	30	30.00	0.00	21
	idpc_ndu_3161_15_339881	30.0	37.5	16.48	30	30.00	0.00	21
	idpc_ndu_3377_17_657018	17.0	25.8	7.68	17	17.00	0.00	11
	idpc_ndu_3602_32_406192	40.0	73.9	12.58	35	38.43	2.21	21
	idpc_ndu_3647_29_457939	30.0	39.7	7.52	30	30.03	0.18	21



(a) RPD on small types



(b) RPD on large types

Fig. 8: The obtained RPD values of all algorithms

VI. Conclusion

This paper first introduced a two-level algorithm based on GA termed PGA to grapple with the IDPC-NDU. In the proposed algorithm, a filtering process and a new chromosome encoding method that decreases the chromosome length to the number of domains are also integrated. As a result, PGA not only improved running time but also reduced resource

TABLE III: The obtained results of all algorithms in Type 2

	Instances	S-ACO			PGA			OPT
		BF	AVG	STD	BF	AVG	STD	
Type 2 Small	idpc_ndu_1002_32_60942	37.0	66.2	9.96	19	19.57	0.90	13
	idpc_ndu_102_10_834	16.0	41.9	28.16	7	7.00	0.00	7
	idpc_ndu_1102_17_56280	50.0	89.0	18.81	25	25.57	1.17	14
	idpc_ndu_1202_18_68002	51.0	91.7	17.75	24	24.70	0.95	15
	idpc_ndu_1302_22_78953	36.0	86.3	20.86	25	25.90	0.31	17
	idpc_ndu_1402_24_92365	65.0	94.3	16.38	24	24.77	0.43	16
	idpc_ndu_1502_27_104878	46.0	86.3	17.02	27	29.00	1.36	16
	idpc_ndu_152_14_1869	16.0	33.0	9.51	16	16.00	0.00	8
	idpc_ndu_1602_14_96765	42.0	122.8	30.42	32	33.73	1.20	17
	idpc_ndu_1702_18_110993	71.0	112.8	19.34	29	31.07	2.29	20
	idpc_ndu_1802_21_123666	73.0	116.0	20.56	34	36.33	0.80	21
	idpc_ndu_1902_27_137407	72.0	114.9	21.02	30	35.10	4.97	21
	idpc_ndu_202_22_2341	20.0	42.9	13.12	20	21.87	2.96	9
	idpc_ndu_252_11_3513	28.0	68.2	26.27	11	17.93	6.60	11
	idpc_ndu_302_12_4930	28.0	66.9	17.88	11	19.23	6.37	11
	idpc_ndu_352_17_6667	34.0	66.0	17.04	28	30.43	0.94	13
	idpc_ndu_402_22_8220	32.0	50.4	16.56	26	26.70	1.37	13
	idpc_ndu_452_32_10406	23.0	74.6	25.64	22	22.00	0.00	13
	idpc_ndu_502_12_10949	49.0	99.5	33.72	11	26.00	6.82	7
	idpc_ndu_52_6_204	6.0	29.1	13.53	6	6.00	0.00	6
Type 2 Large	idpc_ndu_2002_17_128021	82.0	130.6	20.90	37	42.00	2.35	16
	idpc_ndu_2102_19_141155	69.0	150.1	32.51	36	36.00	0.00	20
	idpc_ndu_2202_22_153764	56.0	137.0	35.38	36	36.43	0.50	21
	idpc_ndu_2302_27_169859	65.0	133.1	31.07	42	53.93	8.91	22
	idpc_ndu_2402_29_186438	74.0	144.0	39.90	37	41.10	5.43	23
	idpc_ndu_2502_32_201852	104.0	155.6	33.59	34	54.93	12.42	25
	idpc_ndu_2602_19_185818	113.0	220.1	55.47	41	46.47	3.98	21
	idpc_ndu_2802_30_215589	103.0	182.0	39.99	50	71.60	9.92	25
	idpc_ndu_2902_32_229375	81.0	163.0	29.55	42	51.93	10.05	31

TABLE IV: The statistical values by Wilcoxon signed-rank test with $\alpha = 0.05$ and OPT is the optimal solution

Algorithms	Better	Equal	Worse	p-value	Decision
PGA vs 1.5-OPT	441	97	1142	**0.000**	-
PGA vs 2-OPT	1240	78	362	**0.000**	+
PGA vs 2.5-OPT	1555	0	125	**0.000**	+
PGA vs 3-OPT	1608	27	45	**0.000**	+
PGA vs S-ACO	1451	229	0	**0.000**	+

consumption. Experiments and comparisons with several algorithms on various-sized data sets were conducted to evaluate the proposal's efficiency. The computational results illustrated that PGA outperforms other algorithms and does not exceed twice the optimal values in most instances.

In the future, more and advanced methods will also be adopted to surmount the IDPC-NDU such as the MFEA, and order-based local search algorithms.

ACKNOWLEDGMENT

This research was sponsored by the U.S. Army Combat Capabilities Development Command (CCDC) Pacific and CCDC Army Research Laboratory (ARL) under Contract Number W90GQZ-93290007 for Huynh Thi Thanh Binh. Ta Bao Thang was funded by Vingroup Joint Stock Company and supported by the Domestic Master/ PhD Scholarship Programme of Vingroup Innovation Foundation (VINIF), Vingroup Big Data Institute (VINBIGDATA), code VINIF.2020.ThS.BK.01.

REFERENCES

[1] A. Farrel, J.-P. Vasseur, and J. Ash, "A path computation element (pce)-based architecture," 2006.

[2] D. King and A. Farrel, "The application of the path computation element architecture to the determination of a sequence of domains in mpls and gmpls," *IETF RFC 6805*, 2012.

[3] F. Paolucci, F. Cugini, A. Giorgetti, N. Sambo, and P. Castoldi, "A survey on the path computation element (pce) architecture," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1819–1841, 2013.

[4] L. Maggi, J. Leguay, J. Cohen, and P. Medagliani, "Domain clustering for inter-domain path computation speed-up," *Networks*, vol. 71, no. 3, pp. 252–270, 2018.

[5] P. D. Thanh, H. T. T. Binh, and B. T. Lam, "A survey on hybridizing genetic algorithm with dynamic programming for solving the traveling salesman problem," in *2013 International Conference on Soft Computing and Pattern Recognition (SoCPaR)*. IEEE, 2013, pp. 66–71.

[6] ——, "New mechanism of combination crossover operators in genetic algorithm for solving the traveling salesman problem," in *Knowledge and Systems Engineering*. Springer, 2015, pp. 367–379.

[7] L. D. P. Pugliese and F. Guerriero, "A survey of resource constrained shortest path problems: Exact solution approaches," *Networks*, vol. 62, no. 3, pp. 183–200, 2013.

[8] M. Horváth and T. Kis, "Solving resource constrained shortest path problems with lp-based methods," *Computers & Operations Research*, vol. 73, pp. 150–164, 2016.

[9] L. D. P. Pugliese and F. Guerriero, "Shortest path problem with forbidden paths: The elementary version," *European Journal of Operational Research*, vol. 227, no. 2, pp. 254–267, 2013.

[10] D. Sudholt and C. Thyssen, "A simple ant colony optimizer for stochastic shortest path problems," *Algorithmica*, vol. 64, no. 4, pp. 643–672, 2012.

[11] ——, "Running time analysis of ant colony optimization for shortest path problems," *Journal of Discrete Algorithms*, vol. 10, pp. 165–180, 2012.

[12] A. W. Mohemmed, N. C. Sahoo, and T. K. Geok, "Solving shortest path problem using particle swarm optimization," *Applied Soft Computing*, vol. 8, no. 4, pp. 1643–1653, 2008.

[13] H. T. T. Binh, P. D. Thanh, and T. B. Thang, "New approach to solving the clustered shortest-path tree problem based on reducing the search space of evolutionary algorithm," *Knowledge-Based Systems*, vol. 180, pp. 12–25, 2019.

[14] H. T. T. Binh, T. B. Thang, N. D. Thai, and P. D. Thanh, "A bi-level encoding scheme for the clustered shortest-path tree problem in multifactorial optimization," *Engineering Applications of Artificial Intelligence*, vol. 100, p. 104187, 2021.

[15] H. T. T. Binh, T. B. Thangy, N. B. Long, N. V. Hoang, and P. D. Thanh, "Multifactorial evolutionary algorithm for inter-domain path computation under domain uniqueness constraint," in *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2020, pp. 1–8.