Computing and Informatics, Vol. 42, 2023, 98–125, doi: 10.31577/cai_2023_1_98

ADAPTIVE EVOLUTIONARY MULTITASKING TO SOLVE INTER-DOMAIN PATH COMPUTATION UNDER NODE-DEFINED DOMAIN UNIQUENESS CONSTRAINT: NEW SOLUTION ENCODING SCHEME

Thanh Pham Dinh

Faculty of Natural Sciences and Technology
Tay Bac University
Son La, Vietnam
e-mail: thanhpd@utb.edu.vn, thanhpd05@gmail.com

Abstract. In multi-domain networks, the efficiency of path computation becomes more and more important. The Inter-Domain Path Computation under Node-defined Domain Uniqueness Constraint (IDPC-NDU) is a recently investigated problem where its objective is to determine the effective routing path between two nodes that traverses every domain at most once. IDPC-NDU is NP-Hard, so the approximation approaches are suitable to deal with this problem for large instances. Multifactorial Evolutionary Algorithm (MFEA) is an emerging research topic in the field of evolutionary computation that can efficiently tackle multiple optimization problems at the same time. This study proposed an approach based on the combination of the Adaptive Multifactorial Evolutionary Algorithm (dMFEA-II) and Dijkstra algorithm for solving IDPC-NDU. The encoding and evaluating methods based on the permutation representation are also introduced, and the new individual representation is always to produce valid solutions. The proposed algorithm is evaluated on two types of instances. Simulation results demonstrate the superior performance of the proposed algorithm in comparison with the existing algorithms in terms of the quality of the solution.

Keywords: Adaptive multifactorial evolutionary algorithm, inter-domain path computation, transfer optimization, evolutionary multitasking

1 INTRODUCTION

Nowadays, in the technology era, many devices need to be linked to services or other devices through networks. This will lead to the formation of extremely large networks as well as challenges in finding effective communication methods among devices. To overcome these challenges, large networks are often divided into numerous domains (multi-domain networks). The multi-domain networks also help to tackle issues related to scalability and privacy [1].

In multi-domain networks, Path Computation Element (PCE) is used to compute the path inside each domain. To exchange information among domains, PCE must be communicated together according to a certain architecture. There are two main architectures for handling communication among different PCEs, hierarchical and distributed [2] in which the hierarchical PCE architectures has been promoted in the last few years [3]. The parent PCE is one of the most popular architecture of the hierarchical PCE. To compute an inter-domain path, the parent PCE collects the information of intra-domain routing from all children PCEs.

In [4], the authors introduced a novel domain clustering concept that artificially reduces the number of domains. This concept is expressed as the Inter-Domain Path Computation under Domain Uniqueness constraint (IDPC-DU) problem. In the IDPC-DU, the inter-domain path must satisfy the domain constraint (called the Domain Uniqueness (DU)). This constrain requests that each domain is traversed at most once. There are two variants of the IDPC-DU: the first variant considers the edge set (called the Inter-Domain Path Computation under Edge-defined Domain Uniqueness Constraint (IDPC-EDU)); the second variant considers the vertex set (called the IDPC-NDU). The variants of the IDPC-DU are NP-Hard [4]. In this study, the authors also describe a Dynamic Programming Algorithm (DP) to solve the IDPC-DU in which its computational complexity is $O(|V|^2 2^{|D|}|D|^2)$, where $|V|$ is the number of nodes and $|D|$ is the number of domains. The computational complexity of DP is exponential for the number of domains, so DP is not effective for the multi-domain network consisting of a large number of domains. Therefore, the approximation approach is suitable to tackle the IDPC-DU for the network which includes the large number of domains.

Evolutionary Algorithm (EA) has emerged as an important optimization and search technique in the recent decades [5]. Due to flexible nature and robust behaviour, EA becomes an efficient approach and widely solves global optimization problems. It can be used successfully in various applications of high complexity [6, 7].

In the recent years, MFEA, a new variant of EA, has attracted attention in the research community [8, 9, 10]. The major difference between MFEA and the traditional EA is that the traditional EA solves only one problem in a single run, while MFEA can solve multiple optimization problems simultaneously.

To accelerate convergence and improve the quality of obtained solutions, MFEA takes the advantage of the process of explicit or implicit knowledge transfer across optimization problems [11]. Due to its strong search capability and parallelism,

MFEA is a search technique widely used to find the approximate solutions of optimization problems [12, 13, 14].

In 2019, Bali et al. [9] proposed a new version of MFEA, Multifactorial evolutionary algorithm with online transfer parameter estimation (MFEA-II). MFEA-II overcomes the shortcoming of knowledge transfer process in MFEA by using a dynamic strategy to control the extent of knowledge exchange across problems. In 2020, Osaba et al. [15] introduced another variant of MFEA for discrete optimization problems, dMFEA-II. The paper described a new dynamic mechanism to update the matrix of the exchange of knowledge between problems and a new parent-centric crossover operator for permutation representation.

To take advantage of EA, MFEA, some researchers have implemented EA and MFEA to tackle the variants of IDPC-DU. In [16], the authors proposed a two-level strategy based on evolutionary algorithm (TLGA). An individual in TLGA contains two arrays of genes, so TLGA must get information from both two arrays to construct a solution of the IDPC-NDU. Therefore, TLGA wastes memory and increases the computational complexity for constructing a solution. The solution encoding also leads to the computational complexity of evolutionary operators if TLGA is larger. To deal with the IDPC-NDU, Binh et al. [17] introduced a Two-level Genetic Algorithm (PGA) which divides the original problem into two sub-problems: the first sub-problem finds the order of the visited domains; the second sub-problem finds the shortest path tree [18, 19] from the source node to the destination node. Because the individual representation in PGA is the priority-based encoding and TLGA is focused on creating a valid solution, the quality of the obtained solution may not be improved. The common point of the two algorithms, TLGA and PGA, is that these two algorithms only consider one edge connecting two adjacent regions according to the coding order of the individual. To take advantage of the implicit knowledge transfer for solving the IDPC-EDU, the authors proposed MFEA [11] with a new encoding, many-to-one path encoding. The PMX crossover operator [5] is used for the crossover, and the mutation is a combination of the swap mutation [5] and the one-point mutation [5]. Although experimental results have proven the efficiency of the proposed algorithm, the proposed algorithm still has limitations due to the process of negative transfers. Therefore, to improve the quality of the solution of the IDPC-DU problem, it is necessary to exploit the online transfer parameter estimation to dynamically control the extent of knowledge exchange across tasks in MFEA-II [9].

In order to take the advantages of the dMFEA-II [15] and overcome the disadvantage of existing algorithms for solving IDPC-NDU, this study describes new approach based on dMFEA-II to solve the IDPC-NDU. The main contributions of the proposed algorithm are:

- An individual encoding based on the permutation representation.
- A method for constructing and evaluating the IDPC-NDU solution.
- An approach based on a combination between dMFEA-II and the Dijkstra algorithm.

- The proposed algorithm can use existing evolutionary operators, so it can be implemented easily.
- The experimental results obtained from the proposed algorithm are analyzed on various test instances.

This paper is organized as follows. Section 2 provides the definition of IDPC-NDU. Section 3 brings preliminaries. Section 4 presents the proposed algorithm. Section 5 illustrates and discusses experimental results. Section 6 concludes the paper with discussions about the future extension of this research.

2 PROBLEM DEFINITION AND NOTATIONS

Let $G(V, E, C)$ be a weighted directed multi-graph, where E and V are the set of edges and the set of nodes, respectively. Let $(i, j)^k \in E$ denote the k^{th} parallel edge between nodes i and j. Each edge $(i, j)^k$ has an associated positive weight $w_{i,j}^k$. The set of nodes V is partitioned into H separate clusters (interpreted as domains on the network) $C = \{C^1, C^2, \ldots, C^H\}$. Two nodes, s and $t \in V$ are a source node and a target node, respectively. The objective of the IDPC-NDU is to find a minimum cost path p from the source node s to the target node t that satisfies the Node-defined Domain Uniqueness (NDU) constraint, which allows p to traverse every domain at most once. The IDPC-NDU is stated as follows:

Input:

- A weighted directed multi-graph $G = (V, E, C)$.
- V is partitioned into H domains $C = \{C^1, C^2, \ldots, C^H\}$, $C^i \cap C^j = \emptyset, \forall i, j \in \{1, \ldots, H\}, i \neq j$.
- A source node $s \in V$.
- A destination node $t \in V$.

Output:
A path $p = (p_0, p_1, \ldots, p_l)$ where $p_0 = s$ and $p_l = t$.

Constraint:
A path p has left the domain, it does not revisit it later on. It means that if $p_i \in C^d$ and $p_{i+1} \notin C^d$, then $p_{i+j} \notin C^d, \forall j \geq 2, d \in \{1, \ldots, H\}$.

Objective:

$$f(p) = \sum_{i=0}^{l-1} w(p_i, p_{i+1}) \rightarrow \min.$$

Figure 1 illustrates the definition of the IDPC-NDU in which Figure 1 a) represents an input graph with 7 nodes and 6 domains (each colour presents a domain); the number on each edge is its weight; and the source node and destination node are s and t, respectively. Figure 1 b) depicts a valid path $p_1 = (s, 1, 4, 5, t)$ of the

IDPC-NDU with the cost $f(p_1) = 28$. Figure 1 c) sketches another path from the source node s to the destination node t, $p_2 = (s, 1, 2, 4, 5, t)$. The path p_2 visits the yellow domain at node 1 then enters the violet domain at node 2 and revisits the yellow domain at node 5 so p_2 violates the NDU constraint (it re-enters the yellow domain the second time). Figure 1 d) shows an optimal solution of the IDPC-NDU with the path $p_3 = (s, 4, 5, 3, t)$ and the cost $f(p_3) = 21$.

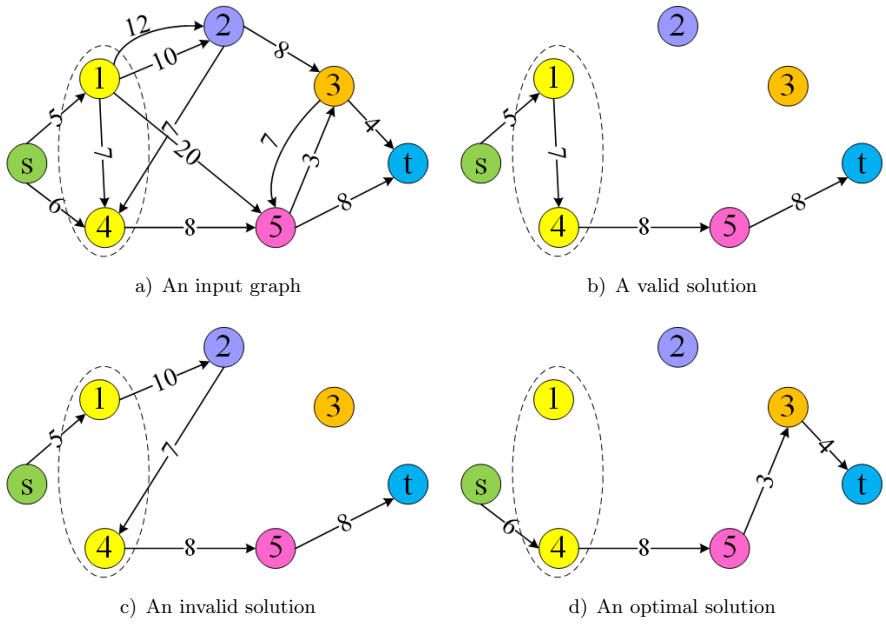


Figure 1. An illustration of solutions of the IDPC-NDU

3 PRELIMINARIES

This section introduces the basic of MFEA and MFEA-II.

3.1 Multitasking Optimization

The main driving force behind Multitasking Optimization (MTO) is to take use of the inter-task synergy to enhance problem solving. Unlike classical EA, MTO solves multiple tasks within only a single task. For the sake of brevity, consider K distinct minimization tasks $\{T_1, T_2, \ldots, T_K\}$ are solved simultaneously.

The objective function of the j^{th} task, denoted T_j is defined as $f_j(x) : X_j \to R$. MTO searches the space of all optimization tasks concurrently for $\{x_1^*, x_2^*, \ldots, x_K^*\} =$

$\text{argmin}\{f_1(x_1), f_2(x_2), \ldots, f_K(x_K)\}$, where $x_j^*, j = 1, \ldots, K$ is a feasible solution in decision space X_j.

To evaluate an individual, Gupta et al. [20] define the properties for each member p_i in a population P as follows:

Factorial Cost: The factorial cost ψ_j^i of an individual p_i on optimization task T_j is defined as $\psi_j^i = \gamma \times \delta_j^i + f_j^i$, where f_j^i and δ_j^i are the objective value and the total constraint violation of p_i, respectively. The coefficient γ is a large penalizing multiplier.

Factorial rank: For an optimization task T_j, the population individuals are sorted in ascending order with respect to the factorial cost. Factorial rank r_j^i of an individual p_i is the index value of p_i in the sort list, relative to all other individuals in the population P.

Scalar Fitness: Scalar fitness φ_i of an individual p_i is based on its best rank over all tasks; i.e. $\varphi_i = 1/\min\{r_1^i, r_2^i, \ldots, r_K^i\}$.

Skill Factor: Skill factor τ_i of an individual p_i is the component task, amongst all other tasks in MTO, with which the individual is associated. This may be defined as $\tau_i = \text{argmin}_{j \in \{1,\ldots,K\}}\{r_j^i\}$.

3.2 Multifactorial Evolutionary Algorithm

In [20], Gupta et al. introduced Multifactorial Optimization (MFO) as an evolutionary multi-tasking paradigm that optimizes multiple tasks simultaneously. In MFO, the individual are encoded in a Unified Search Space (USS), denoted X encompassing $X_1, X_2, \ldots, X_K$, and can decoded into a task-specific solution representation with respect to each of the K optimization tasks.

The basic structure of the MFEA is presented in Algorithm 1 where RMP is a prescribed random mating probability [15, 9].

3.3 An Adaptive Multifactorial Evolutionary Algorithm for Permutation-Based Discrete Optimization Problems

The initial phases of dMFEA-II [9, 15] are the same as those in MFEA. The main differential factor between dMFEA-II and MFEA is the incorporation of the online RMP learning module. In dMFEA-II, RMP is a symmetric $K \times K$ matrix with element's values in the range [0.0, 1.0]. $RMP_{i,j}$ indicates the probability of conducting an inter-task crossover between i^{th} and j^{th} tasks. Two parameters Δ_{inc} and Δ_{dec} are used for determining the evolution of each $RMP_{i,j}$ as follow:

- RMP_{τ_i,τ_j} is incremented using Δ_{inc} as $RMP_{\tau_i,\tau_j} = \min\{1.0, RMP_{\tau_i,\tau_j}/\Delta_{inc}\}$,
- RMP_{τ_i,τ_j} is decreased using Δ_{dec} as $RMP_{\tau_i,\tau_j} = \max\{1.0, RMP_{\tau_i,\tau_j} \times \Delta_{inc}\}$.

To adapt OX crossover [5] to the parent-centric feature [9], dMFEA-II limits the size of the cutting windows (the segment of the individual lying two random cutting

Algorithm 1: Basic structure of the MFEA

```
1  begin
2      Randomly sample N individuals in X to form initial population P(0);
3      for every individual p_i in P(0) do
4          Assign skill factor τ_i =  mod (i, K) + 1, for the case of K tasks;
5          Evaluate p_i for task τ_i only;
6      t ← 1;
7      while stopping conditions are not satisfied do
8          Configure offspring population P_c(t) = ∅;
9          while offspring generated for each task < N do
10             Sample two individuals uniformly at random (without
                replacement): x_i and x_j from P(t);
11             if τ_i = τ_j then
12                 [x_a, x_b] ← Intra-task crossover between x_i and x_j;
13                 Assign offspring x_a and x_b skill factor τ_i;
14             else
15                 if rand < RMP then
16                     [x_a, x_b] ← Inter-task crossover between x_i and x_j;
17                     Each offspring is randomly assigned skill factor τ_i or τ_j;
18                 else
19                     [x_a] ← local variantion (mutation) of x_i;
20                     Assign offspring x_a skill factor τ_i;
21                     [x_b] ← local variantion (mutation) of x_j;
22                     Assign offspring x_b skill factor τ_j;
23             Evaluate [x_a, x_b] for their assigned skill factors only;
24             P_c(t) ← P_c(t) ∪ [x_a, x_b];
25         P ← Select the best N individuals in P ∪ P_c(t) as per their scalar
              fitness;
26         t ← t + 1;
27     return The best individuals in P for each task T_k;
```

points) to a fraction $W \in [0, 1]$ of the total dimension as $W \times RMP_{i,j} \times D_i$ where D_i is the dimensionality of i^{th} task. The mutation parameter $P_m \in [0, 1]$ in dMFEA-II controls whether a new individual x_a or x_b should perform mutation.

dMFEA-II is obtained from MFEA in Algorithm 1 by replacing lines 15 – 22 with those in Algorithm 2.

4 PROPOSED ALGORITHM

This section describes our novel approach for solving the IDPC-NDU (N-dMFEA-II). The proposed algorithm includes: a new individual encoding, a new method for constructing the solution, a decoding method, and evolutionary operators.

Algorithm 2: Crossover strategy of dMFEA-II

begin
  if $\tau_i \neq \tau_j$ **then**
    if *rand1* $\leq RMP_{\tau_i,\tau_j}$ **then**
      $[x_a, x_b] \leftarrow$ Inter-task crossover between x_i and x_j;
      if *rand2* $\leq p_m$ **then**
        $x_a \leftarrow$ mutation(x_a);
        $x_b \leftarrow$ mutation(x_b);
      Each offspring is randomly assigned skill factor τ_i or τ_j;
      Update RMP_{τ_i,τ_j} using Δ_{inc} and Δ_{dec};
    else
      Randomly select x_{p_1} with $\tau_{p_1} = \tau_i$ and $p_1 \neq i$;
      $x_a \leftarrow$ Intra-task crossover between x_i and x_j;
      if *rand2* $\leq p_m$ **then**
        $x_a \leftarrow$ mutation(x_a);
      Assign offspring x_a skill factor τ_i;
      Update RMP_{τ_i,τ_i}
      Randomly select x_{p_2} with $\tau_{p_2} = \tau_j$ and $p_2 \neq j$;
      $x_b \leftarrow$ Intra-task crossover between x_{p_2} and x_j;
      if *rand2* $\leq p_m$ **then**
        $x_b \leftarrow$ mutation(x_b);
      Assign offspring x_b skill factor τ_j;
      Update RMP_{τ_j,τ_j}
  return The best individuals in P for each task T_k;

4.1 Motivation of Proposed Algorithm

In the previous studies [16, 17], the major steps for constructing a solution of IDPC-NDU are as follows:

Step 1: Generate an order of domains that the path will go through.

Step 2: Construct a graph according to an order of domains in which an inter-domain edge between two nodes is only created if the corresponding domains are adjacent.

Step 3: Find the shortest path from the source node to the destination node in the constructed graph in Step 2.

The strength of these approaches is that if a solution is produced by the approach then it is valid. However, it has a common drawback, i.e., the previous studies only create inter-domain edges connecting two adjacent domains, but in some cases, the inter-domain edges connecting two non-adjacent domains can improve the quality of solution. Figure 2 illustrates an example of the shortcoming of the existing individual representation with the input graph as in Figure 1 a). Suppose that the

order of domains is $Green \rightarrow Yellow \rightarrow Violet \rightarrow Orange \rightarrow Pink \rightarrow Blue$. After performing step 2, a new graph is constructed according to the order of the domain as in Figure 2 a). The shortest path from s to t of the input graph in Figure 2 a) is $p_1 = (s, 1, 2, 3, 5, t)$ with the cost of 38 (Figure 2 b)). If there is an inter-domain edge connecting two non-adjacent domains, *orange* and *blue* domains (the dotted edge $(3, t)$ in Figure 2 c)) then the shortest path is $p_2 = (s, 1, 2, 3, t)$ with the cost of 27 (Figure 2 d)). The cost of the path p_2 is smaller than the one of the path p_1. It means that if inter-domain edges connecting non-adjacent domains can help to decrease the cost of the IDPC-NDU solution.

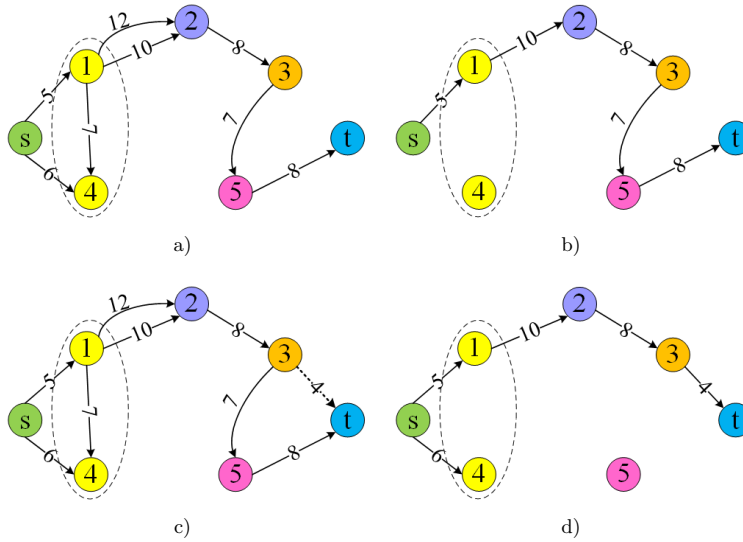


Figure 2. The disadvantage of the existing individual representations

To overcome these disadvantages, in this study, an individual encoding based on permutation representation is introduced. The new individual encoding builds inter-domain edges which link non-adjacent domains.

4.2 Structure of the Proposed dMFEA-II

The pseudocode of the proposed approach is presented in Algorithm 3 where K, D_i is the number of tasks and the number of domains in i^{th} task, respectively. It should be noted that the cost of an individual is computed through the cost of the corresponding solution of the IDPC-NDU. It means that to evaluate an individual p_i, N-dMFEA-II builds and computes the cost of the corresponding IDPC-NDU solution s_i and then the cost of the individual p_i is computed.

The solution of the IDPC-NDU can be constructed through the order of the domains, so a chromosome in N-dMFEA-II only needs to store the order of domains.

Algorithm 3: New approach to solve the IDPC-NDU

Input:
– The weighted directed multi-graphs $G_i = \{V_i, E_i, C_i\}, i \in 1, \ldots, K$; $C_i = \{C_i^1, C_i^2, \ldots, C_i^{D_i}\}$;
– The source nodes $s_i \in V_i$;
– The destination nodes $t_i \in V_i$;
Output: IDPC-NDU solutions;

begin
  $t \leftarrow 0$;
  $P_t \leftarrow$ Randomly generate N individuals;
  foreach *individual* $ind_j \in P_t$ **do**
    Assign random skill factor τ_j for ind_j;
    Construct solution s_j of IDPC-NDU based on the individual ind_j and τ_j ▷ Refer to Subsection 4.6;
    Compute the factorial cost of the solution s_j;
    Compute the factorial rank, the scalar fitness of the individual ind_j;
  $t \leftarrow 0$;
  while *stopping conditions are not satisfied* **do**
    $O_t \leftarrow \emptyset$;
    $P_t^{'} \leftarrow$ Tournament Selection(P_t);
    $O_t \leftarrow$ Perform crossover and mutation operator$(P_t^{'})$;
    foreach *individual* $o_j \in O_t$ **do**
      Construct solution $s_j^{'}$ of IDPC-NDU based on its assigned skill factor and the individual o_j ▷ Refer to Subsection 4.6;
      Evaluate the factorial cost of o_j based on the cost of solution $s_j^{'}$;
    $R_t \leftarrow O_t \cup P_t$;
    Update the factorial ranks, scalar fitness and skill factor of every individuals in R_t;
    $P_{t+1} \leftarrow$ Select N fittest members from R_t;
    $t \leftarrow t + 1$;
  return The best solution of IDPC-NDU in each task;

Therefore, to evaluate an individual, N-dMFEA-II performs the decoding method to build an individual for each task, then it constructs the unweighted directed graph G' (*also known as an H-Graph, where "H" implies a graph where each vertex corresponds to a domain of the input graph*) based on the received individual. After that, a weighted directed graph G'' (*called an L-Graph, where "L" refers to a graph where each vertex corresponds to a node in the input graph*) is constructed from the H-Graph G' and the input graph G, and then finds the shortest path from the source node to the destination node in the L-Graph G''.

4.3 Individual Representation

4.3.1 Individual Representation for a Task

In N-dMFEA-II, a solution of the IDPC-NDU can be encoded by an order of the domains. Therefore, if the input graph of a i^{th} task has D_i domains then an individual in this task is a permutation of the set $\{1, \ldots, D_i\}$.

Figure 3 shows an example of individual representation for a task in N-dMFEA-II. Suppose that the input graph including 7 nodes and 6 domains is sketched as in Figure 1 a) and the order of domains is *Green* $\rightarrow$ *Yellow* $\rightarrow$ *Violet* $\rightarrow$ *Orange* $\rightarrow$ *Pink* $\rightarrow$ *Blue*. As a result, the order can be encoded to an individual as shown in Figure 3 a). For the sake of simplicity, the domain colors are labeled as numbers, i.e., in Figure 3 a), the Blue, Green, Violet, Orange, Pink, and Yellow domains are labelled 1, 2, 3, 4, 5, and 6, respectively. Consequently, the individual in Figure 3 a) is depicted as shown in Figure 3 b).

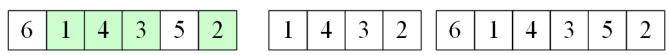
a) b)

Figure 3. An illustration of individual encoding method for a task

This individual representation helps to reduce the size of storage memory and computational complexity in comparison with the individual encoding by a path of nodes in the graph. Another benefit of this individual representation is that the IDPC-NDU solution created from an individual always satisfies NDU constraint.

4.3.2 Individual Representation in Unified Search Space

Because the solution of a task is encoded by the permutation representation [5], N-dMFEA-II uses the permutation representation for encoding individual in USS [21, 22]. Suppose that the K task of IDPC-NDUs are solved and D_i is the number of domains in i^{th} task. Then the dimension of chromosome in USS is $D_{USS} = \max_{i \in \{1,\ldots,K\}} D_i$.



a) b) c)

Figure 4. An example of individual representation in USS

Figure 4 depicts an example of individual representation in USS for two IDPC-NDU tasks including 4 and 6 domains respectively. Figure 4 a) illustrates a chromosome in USS while Figure 4 b) and Figure 4 c) sketch the chromosomes of two tasks which are associated with the chromosome in Figure 4 a).

4.4 Evolutionary Operators

A major difference between MFEA and MFEA-II is that the crossover operator in MFEA-II has parent-centric characteristic [15]. In N-dMFEA-II, the inter-task crossover and the intra-task crossover are Dynamic Order Crossover (dOX) [15] and Order Crossover (OX) [7], respectively. The mutation operator is 2-opt [23, 24].

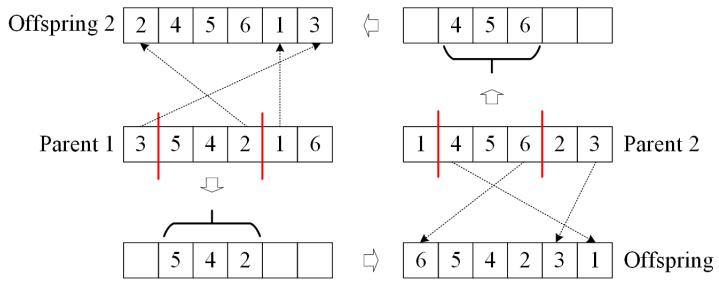


Figure 5. An example of the Order crossover operator

Figure 5 illustrates OX in which two vertical red lines present two random crossover points. For creating the offspring 1, the segment (cutting window) between two crossover points (including genes 5, 4 and 2) is copied from the first parent. After that, the remaining unused genes from the second crossover point in the second parent, i.e., 3, 1, and 6, are copied into this offspring. The second offspring is created in an analogous manner, with the parent roles reversed.

dOX is a modified version of OX in which OX is edited to adapt to the parent-centric feature [9, 15] by limiting the size of the cutting window to $W \times RMP_{k,k'} \times D_k$, where $W \in [0, 1]$ is a parameter; $RMP_{k,k'}$ is the probability of conducting an inter-task crossover between tasks k and k'; and D_k is the dimensionality of task T_k.

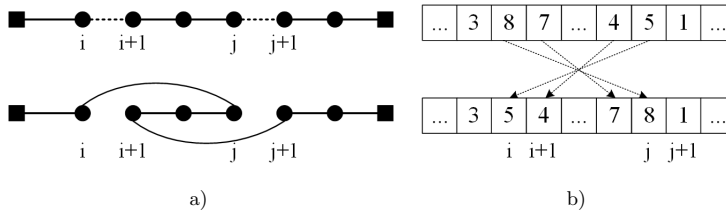


Figure 6. An example of the 2-opt mutation operator

Figure 6 depicts an example of the 2-opt mutation operator, which removes two edges and reconnects the paths created. Figure 6 a) illustrates the path of an individual before (the above part) and after (the below part) performing this mutation operator, where the dashed lines are edges to be deleted; the arc edges are

edges to be added. Figure 6 b) illustrates the position of genes of an individual before (the above part) and after (the below part) performing this mutation operator.

4.5 Decoding Method

To find an individual ind' for i^{th} task, T_i from an individual ind_U in USS, N-dMFEA-II selects all the integers no larger than D_i from ind_U and keeps them in the same relative order as in ind_U.

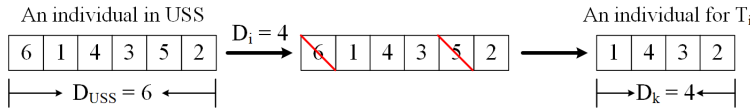


Figure 7. An illustration of the decoding method in N-dMFEA-II

Figure 7 presents an example of a decoding method from an individual in USS to a task in which the dimension of the individual in USS and the dimension of the task are 6 and 4, respectively. Because ind_U has two genes larger than $D_i = 4$, these genes are not selected. The remaining genes, i.e., 1, 4, 3, 2 are selected to create an individual for the task.

4.6 Solution Construction and Evaluation

Because each individual $ind_t = (ind_t^1, ind_t^2, \ldots, ind_t^{D_t})$ (D_t is the number of domains in the input graph of the t^{th} task) in N-dMFEA-II is a permutation of the set of domains, to construct a solution of the IDPC-NDU from an individual ind_i, N-dMFEA-II finds the shortest path from the source node to the destination node according to the order of domains in ind_t. Considering the input graph G, the cost of an individual ind_t is computed in four steps:

Step 1: Determine the order of domains based on the order of genes in ind_i.

Step 2: Construct a H-Graph $G' = (V', E')$ according to the order of domains as follows:

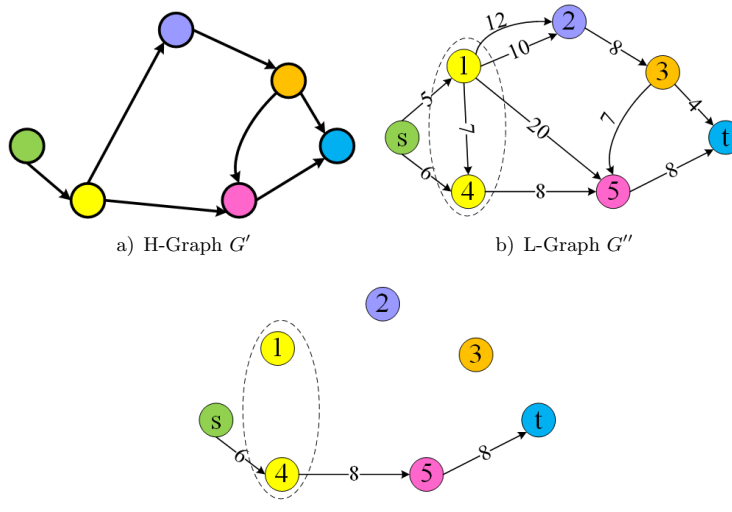
- A vertex $v'_i \in V'$ is a representation of a domain ind_t^i.
- There is a directed edge $e' = (v'_i, v'_j) \in E'$ connecting from node $v'_i \in V'$ to vertex $v'_j \in V'$ when there is at least one edge in G connecting a node in domain ind_t^i to a vertex in domain ind_t^j and $i < j$.

Step 3: Construct a L-Graph G'' based on G and G' in the following steps:

- G'' and G are the same vertex set.
- For inter-domain edges: edges connecting from domain ind_t^j to domain ind_t^{j+1} ($j = 1, \ldots, D_t - 1$) or connecting from domain $ind_t^{D_t}$ to domain ind_t^1 are kept, other inter-domains edges are removed.

- For intra-domain edges: all edges in G are retained.

Step 4: Perform the Dijkstra algorithm [25, 26] for finding the shortest path from the source node to the destination node in G''.



a) H-Graph G' b) L-Graph G''

c) A solution of IDPC-NDU

Figure 8. An illustration of the individual evaluation method in N-dMFEA-II

The main steps of the individual evaluation method are sketched in Figure 8. Figure 8 a) depicts an H-Graph G' which is constructed based on the input graph G in Figure 1 a) and an individual in Figure 3. Because the *yellow* domain is before the *violet* domain in G' and there is an edge (1, 2) in G connecting these two domains, there is an edge connecting the *yellow* and *violet* domains. The other edges of G' are similarly created. Figure 8 b) sketches a L-Graph G'' which is constructed from G and G'. Because inter-domain edges, i.e., $(s, 1)$, $(s, 4)$, $(1, 2)$, $(4, 5)$, $(1, 5)$, $(2, 3)$, $(3, 5)$, $(3, t)$ and $(5, t)$ are the same direction as the edges connecting vertices in G' in Figure 8 a), these edges are preserved and the remaining inter-domain edges are removed. The shortest path from the source node s to the destination node t in G'' is $s \to 4 \to 5 \to t$ (as shown in Figure 8 c)) with the cost of 22.

5 COMPUTATIONAL RESULTS

5.1 Problem Instances

In order to examine the effectiveness of N-dMFEA-II in solving the IDPC-NDU, we select two datasets:

- The Artificially Generated Synthetic Dataset (AGSD) [27] includes two distinct types of instances, which are also used in the previous works [16, 17]. AGSD is created as follows: Firstly, three parameters are generated for each instance: the number of nodes, the number of domains, and the number of edges. Secondly, an optimal path p is created in which the weight of edges is equal to 1 and the number of domains on p is approximately the number of domains of the input graph. The noise is then added to the instance for each 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. Especially in Type 2, feasible paths whose length is less than three are removed. Each type includes two sets regarding dimensionality: a set of small instances with the number of nodes from 50 to 2000, and a set of large instances with the number of nodes over 2000.
- The instances from the Internet Topology Zoo Dataset (ITZD) [28] in which the range of these instances is country+, continent, continent+, or global, and the position of a node in an instance is determined by its longitudes and latitudes, obtained through geocoding of Points-of-Presence (PoP) locations. The type of selected instances is the backbone network and they belong to the commercial network. However, it was necessary to add information about a source and a destination nodes to each of instances. Therefore, for each instance, we randomly select two nodes as the source and destination nodes, which belong to a different country. For each instance, the country property of the nodes plays as the domain, i.e., if two nodes belong to a country, then these nodes are considered to belong to a domain.

A summary of the datasets is shown in Table 1.

Dataset		**NoVertices**	**NoDomains**	**NoEdges**	**NoIns**
AGSD	Minimum	427	7	14 927	16
Type 1 Small	Maximum	2 002	42	178 026	
AGSD	Minimum	2 102	12	164 811	9
Type 1 Large	Maximum	7 352	32	1 461 349	
AGSD	Minimum	52	6	204	20
Type 2 Small	Maximum	1 902	32	137 407	
AGSD	Minimum	2 002	17	128 021	8
Type 2 Large	Maximum	2 802	30	229 375	
ITZD	Minimum	8	2	7	21
	Maximum	93	37	216	

NoVertices: The number of Vertices; **NoDomains**: The number of domains; **NoEdges**: The number of edges; **NoIns**: The number of instances.

Table 1. Summary the types of the IDPC-DU instances

5.2 Experimental Criteria

To make our comparison fair, Table 2 presents criteria for evaluating the quality of the obtained solution from the algorithms.

Average (Avg)	The average function value over all
Best-found (BF)	Best function value achieved over all runs
Std	Standard deviation
NoNF	The number of runs in which an algorithm does not find a solution

Table 2. Criteria for assessing the quality of the output of the algorithms

5.3 Experimental Settings

To evaluate the performance of the proposed algorithm, we analyze the received results of N-dMFEA-II, the two existing algorithms, i.e., TLGA [16] and PGA [17], and the optimal solutions.

The parameters of N-dMFEA-II are set up in a similar experimental environment of dMFEA-II [15] and are summarized in Table 3.

Parameters	**Value**	**Parameters**	**Value**
Population size	100	Initial values of $RMP_{k,k'}$	0.95
Crossover rate	0.95	Intra-task crossover operator	OX [7]
Mutation rate	0.05	Inter-task crossover operator	dOX [15]
$\Delta_{inc}/\Delta_{dec}$	0.99/0.99	Mutation operator	2-opt [15]
		Cutting window size (W)	0.9

Table 3. Parameter values for N-dMFEA-II

This study implements N-dMFEA-II with two tasks. Each task is a different instance of the IDPC-NDU.

In the result tables (Tables 4, 5, and 10), two instances that are in a case are selected to perform N-dMFEA-II simultaneously. If the number of instances is odd, then an instance will be randomly paired with another instance. For example, given a set of instances A, B, C, D, E, F, and G, suppose that the pairs of instances are (A, C), (B, D), and (E, F). Since there is no more unused instance to pair with instance G, one of the instances in the sets $\{A, B, C, D, E, F\}$ is randomly selected to pair with instance G. Suppose that the generated pair is (G, E), because the solution of the instance E has been found in the pair (E, F), so the solution of the instance E in the pair (G, E) is not used.

To make our comparison fair, for each instance, all algorithms are simulated 30 times on the computer with a CPU – Intel Xeon E5620, RAM – 8 GB. The source codes were implemented in the C# language. The total number of objective function evaluations is 50 000.

Dijkstra algorithm [25] is implemented for finding the shortest path.

5.4 Experimental Scenario

Three set of experiments are conducted for assessing the performance of N-dMFEA-II:

- The non-parametric statistic is used for analysing the received results of N-dMFEA-II and the existing algorithms.
- Comparing the results achieved by N-dMFEA-II with the existing algorithms.
- Because the performance of the proposed algorithm was contributed by parameters: the relationship between the number of nodes and the number of domains, etc. is conducted to evaluate the effect of these parameters.

5.5 Experimental Results on the AGSD Dataset

In this subsection, the experimental results obtained by N-dMFEA-II are compared with the ones obtained by the two newest algorithms, PGA [17] and TLGA [16] on instances in the AGSD dataset.

5.5.1 Non-Parametric Statistic for Comparing the Results of the Proposed Algorithm and Existing Algorithms

The non-parametric statistic is used for analyzing the received results of the algorithms. We perform two main steps in the comparison process:

- Firstly, statistical methods such as Friedman, Aligned Friedman, Quade [29, 30] are used to evaluate the differences among results obtained by the aforementioned algorithms.
- Secondly, the hypothesis of equivalence of means of the results obtained by algorithms in the first step is rejected, and then the post-hoc statistical procedures [29, 30] are applied to the obtained results to compute the concrete differences among algorithms and compare the control algorithm with the remaining algorithms.

Tables 4 and 5 present the experimental results obtained by algorithms on the types of instances. In these tables, the italic, red cells in a column of an algorithm denote instances where this algorithm exceeds the others.

The results of Friedman's, Iman-Davenport's, Aligned Friedman's, and Quade's tests are shown in Table 6. As we can see in this table, all values of the Friedman, Iman-Davenport, Aligned Friedman, and Quade tests are greater than their associated critical values. Furthermore, all p-values are less than 0.05, so all the null hypotheses, i.e., the equivalence of the medians of the results of the different benchmarks are rejected. It means that there are significant differences among the observed results with a level of significance $\alpha \leq 0.05$.

Table 7 presents the ranks computed through the Friedman, Friedman Aligned, and Quade procedures. The results in this table point out the existence of significant

	Case	Instance	TLGA			PGA			N-dMFEA-II		
			BF	Avg	Std	BF	Avg	Std	BF	Avg	Std
Small instances	1	idpc_ndu_1002_22_36564	42	42	0.0	30	34	2.9	30	***30***	0.0
		idpc_ndu_1192_19_37744	63	63	0.0	40	47.3	5.1	40	***40.1***	0.5
	2	idpc_ndu_1202_22_65521	22	22.5	0.9	22	***22***	0.0	22	***22***	0.0
		idpc_ndu_1256_21_44446	77	97.1	4.0	46	47.9	2.6	42	***42.3***	0.5
	4	idpc_ndu_1322_22_48821	48	48	0.0	44	44	0.0	26	***42.7***	3.2
		idpc_ndu_1506_9_130556	19	20	1.5	11	12.3	3.0	11	***11***	0.0
	5	idpc_ndu_1514_16_78292	37	50.3	2.6	33	***33***	0.0	33	***33***	0.0
		idpc_ndu_1602_22_60574	52	101.8	13.9	46	46.8	0.4	46	***46***	0.0
	6	idpc_ndu_1514_30_78351	30	***30***	0.0	30	***30***	0.0	30	***30***	0.0
		idpc_ndu_1682_23_67612	78	78	0.0	46	55	11.5	46	***46***	0.0
	7	idpc_ndu_1730_20_88509	42	42	0.0	39	41.8	0.8	39	***39***	0.0
		idpc_ndu_2002_22_178026	24	24.7	4.0	24	***24***	0.0	24	***24***	0.0
	8	idpc_ndu_427_7_14927	13	13	0.0	8	***8***	0.0	8	***8***	0.0
		idpc_ndu_704_15_16990	34	72.6	13.1	31	***31***	0.0	31	***31***	0.0
	9	idpc_ndu_842_23_31617	22	***22***	0.0	22	***22***	0.0	22	***22***	0.0
		idpc_ndu_1002_12_82252	14	14.2	0.5	8	13.8	1.1	8	***8.4***	1.5
Large instances	10	idpc_ndu_2102_23_164811	27	***27***	0.2	27	***27***	0.0	27	***27***	0.0
		idpc_ndu_2402_22_260967	24	25.4	5.0	24	***24***	0.0	24	***24***	0.0
	11	idpc_ndu_2494_12_276620	26	30.8	5.9	23	23	0.0	16	***21.1***	3.2
		idpc_ndu_2502_22_229601	35	36.2	2.7	29	***29***	0.0	29	***29***	0.0
	12	idpc_ndu_2522_20_171940	77	77.3	1.5	41	41	0.0	18	***37.2***	8.7
		idpc_ndu_2715_22_592246	13	13.9	0.8	13	13	0.0	8	***8.8***	1.9
	13	idpc_ndu_2918_29_293109	30	31.5	2.0	30	***30***	0.0	30	***30***	0.0
		idpc_ndu_3161_15_339881	30	41.7	11.9	30	***30***	0.0	30	***30***	0.0
	14	idpc_ndu_3602_32_406192	71	78.3	5.4	35	38.8	2.0	35	***35***	0.0
		idpc_ndu_3602_32_406192	71	78.3	5.4	35	38.8	2.0	35	***35***	0.0

Table 4. Results obtained by TLGA, PGA and N-dMFEA-II on instances in Type 1

differences among the algorithms considered. From this table, TLGA is the worst performing algorithm, whereas N-dMFEA-II is the best. Therefore, N-dMFEA-II is the control algorithm.

Table 8 shows ranks, unadjusted p-value, and z-values which are computed for the algorithms TLGA and PGA. The unadjusted p-values demonstrated that both algorithms TLGA and PGA are significantly worse than the control algorithm at a level of significance $\alpha = 0.05$.

The above mentions lead to the conclusion that, N-dMFEA-II beats both algorithms TLGA and PGA on most cases.

	Case	Instance	TLGA			PGA			N-dMFEA-II		
			BF	Avg	Std	BF	Avg	Std	BF	Avg	Std
Type 2 Small Large instances	1	idpc_ndu_1002_32_60942	19	37.1	3.5	19	19.6	0.9	13	***18.8***	1.1
		idpc_ndu_1102_17_56280	49	57.9	2.7	25	25.6	1.2	25	***25***	0.0
	2	idpc_ndu_52_6_204	6	16.3	11.2	6	***6***	0.0	6	***6***	0.0
		idpc_ndu_102_10_834	19	26.9	5.9	7	***7***	0.0	7	***7***	0.0
	3	idpc_ndu_1202_18_68002	31	34.9	0.7	24	24.7	1.0	15	***23.7***	1.6
		idpc_ndu_1302_22_78953	45	63.2	6.8	25	25.9	0.3	25	***25***	0.0
	4	idpc_ndu_1402_24_92365	41	80.3	9.5	24	24.8	0.4	16	***23.7***	1.5
		idpc_ndu_1502_27_104878	30	57	9.4	27	29	1.3	26	***26***	0.0
	5	idpc_ndu_1602_14_96765	46	80.9	10.0	32	33.7	1.2	17	***18***	3.8
		idpc_ndu_1702_18_110993	40	40	0.0	29	31.1	2.3	20	***20.9***	2.7
	6	idpc_ndu_1802_21_123666	68	70.9	0.6	34	36.3	0.8	33	***33.1***	0.3
		idpc_ndu_1902_27_137407	73	101.1	8.1	30	35.1	5.0	30	***30***	0.0
	7	idpc_ndu_152_14_1869	16	34.8	12.3	16	16	0.0	8	***8***	0.0
		idpc_ndu_202_22_2341	27	31.3	10.3	20	21.9	3.0	9	***16***	5.4
	8	idpc_ndu_252_11_3513	24	41.4	9.9	11	17.9	6.6	11	***11***	0.0
		idpc_ndu_302_12_4930	42	45.9	0.7	11	19.2	6.4	11	***12.7***	4.5
	9	idpc_ndu_352_17_6667	36	46.9	11.3	28	30.4	0.9	13	***19.6***	5.5
		idpc_ndu_402_22_8220	32	51.1	24.3	26	26.7	1.4	13	***25.6***	2.4
	10	idpc_ndu_452_32_10406	22	86.7	32.9	22	22	0.0	13	***21.7***	1.6
		idpc_ndu_502_12_10949	29	54.5	12.0	11	26	6.8	11	***11***	0.0
	11	idpc_ndu_2002_17_128021	78	78	0.0	37	42	2.3	16	***34.2***	7.3
		idpc_ndu_2102_19_141155	96	107.6	3.5	36	***36***	0.0	36	***36***	0.0
	12	idpc_ndu_2202_22_153764	44	88.8	14.1	36	36.4	0.5	35	***35***	0.0
		idpc_ndu_2302_27_169859	76	105.3	6.7	42	53.9	8.9	36	***36.5***	1.1
	13	idpc_ndu_2402_29_186438	124	136.1	3.0	37	41.1	5.4	37	***37***	0.0
		idpc_ndu_2502_32_201852	67	124.7	13.2	34	54.9	12.4	34	***34***	0.0
	14	idpc_ndu_2802_30_215589	79	154.4	22.2	50	71.6	9.9	40	***40.3***	1.0

Table 5. Results obtained by TLGA, PGA and N-dMFEA-II on instances in Type 2

5.5.2 Detail of Comparison Among the Algorithms TLGA, PGA, and N-dMFEA-II

The results obtained by N-dMFEA-II in comparison with two algorithms (PGA and TLGA) are briefed in Table 9. In this table, $A \ll B$ denotes the number of instances on which algorithm A outperforms algorithm B. Similarly, $A \approx B$ denotes the number of instances on which algorithm A is equivalent to algorithm B.

Some comments are worth being mentioned from Table 9:

- N-dMFEA-II surpasses both algorithms PGA and TLGA on almost all cases. More specifically, it is noted that N-dMFEA-II outperforms TLGA on all instances in Type 2 and there is no test case on which PGA or TLGA outperforms N-dMFEA-II.

Friedman Value	X^2 Value	p-Value	I-D Value	F_F Value	p-Value
84.38	5.99	$5.0 * 10^{-11}$	**202.92**	3.08	$1.3 * 10^{-36}$

A. Friedman Value	X^2 Value	p-Value	Quade Value	F_F Value	p-Value
35.66	5.99	$1.8 * 10^{-8}$	**121.01**	3.08	$7.1 * 10^{-28}$

A. Friedman: Aligned Friedman; **I-D**: Iman-Davenport;

Table 6. Results of the Friedman, Iman-Davenport, Aligned Friedman and Quade tests ($\alpha = 0.05$)

Algorithms	Friedman	Friedman Aligned	Quade
N-dMFEA-II	1.1698	46.6038	1.0776
TLGA	2.9434	132.3585	2.9958
PGA	1.8868	61.0377	1.9266

Table 7. Average rankings achiedved by the Friedman, Friedman Aligned, and Quade tests

- N-dMFEA-II beats PGA on 13 of 25 instances on Type 1 (approximation 52 % of total cases) and on 25 of 28 instances on Type 2 (approximation 89 % of total cases). It means that the performance of N-dMFEA-II tends to increase on instances in Type 2 in comparison with PGA.
- There are only 3 test cases where the results obtained by TLGA are equal to the ones obtained by N-dMFEA-II. It leads to the conclusion that the performance of N-dMFEA-II significantly exceeds that of TLGA.

5.5.3 Analysis of Influential Factors

Because the individual encoding depends on the number of domains, this subsection analyzes the influence of the number of domains on the performance of N-dMFEA-II in comparison with the existing algorithms.

To determine the correlation between the number of domains and the results obtained by algorithms, the scatter plots of the relationship between the number of domains and the comparison between N-dMFEA-II with two algorithms TLGA and PGA for each type are plotted. In these figures, the circle symbols mean that the performance of N-dMFEA-II exceeds the comparing algorithms; on the contrary, the

		Friedman				**Quade**			
i	Alg	z	p	Holm	Hol	z	p	Holm	Hol
2	TLGA	9.13	$6.8 * 10^{-20}$	0.025	0.025	8.59	$8.6 * 10^{-18}$	0.025	0.025
1	PGA	3.69	$2.2 * 10^{-4}$	0.05	0.05	3.80	$1.4 * 10^{-4}$	0.05	0.05

Alg: Algorithms; **Hol**: Holland; **z**: z-values; **p**: unadjusted p-values

Table 8. The z-values and p-values of the Friedman, Quade procedures (N-dMFEA-II is the control algorithm)

		N-dMFEA-II				
		≪PGA	≈ PGA	≪ TLGA	≈ TLGA	**Total**
Type 1	Small	9	7	14	2	16
	Large	4	5	8	1	9
Type 2	Small	18	2	20	0	20
	Large	7	1	8	0	8

Table 9. The summary of comparison of achieved results from N-dMFEA-II and the existing algorithms

square symbols represent instances where the performance of N-dMFEA-II is worse than the performance of the comparing algorithms. The triangle symbols mean that the performance of two algorithms is equal.

Figures 9 and 10 illustrate the relationship between the number of domains and the comparative results among TLGA, PGA, and N-dMFEA-II for Type 1 and Type 2, respectively.

It is inferred from Figure 9 that N-dMFEA-II and TLGA are equal on three instances idpc_ndu_2102_23_164811, idpc_ndu_842_23_31617 and idpc_ndu_1514_30_78351. In other words, the proposed scheme performs better than TLGA on instances in which the number of domains is 23 or less. The comparative results between N-dMFEA-II and PGA in Figure 9 b) are different from other cases when square and trial symbols are interspersed. It means that the comparative results are less dependent on the number of domains.

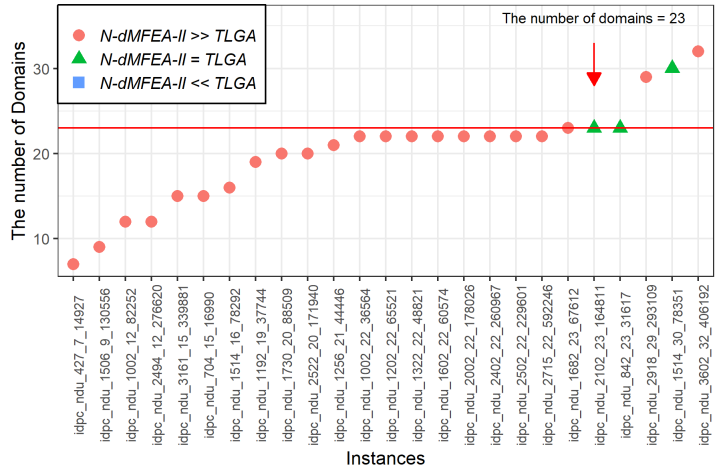
Because N-dMFEA-II outperforms TLGA on all instances in Type 2, only the graph of comparison between N-dMFEA-II and PGA is plotted in this type. As can be observed from the results in Figure 10, the two algorithms are equal on instances, in which the number of domains is 19 or less; for larger instances, N-dMFEA-II outperforms PGA all test cases. It implies that the performance of N-dMFEA-II tends to be better than that of PGA when the number of domains increases.

According to the above mentions, the performance of N-dMFEA-II tends to increase when the number of domains decreases (for instances in Type 1) or the number of domains increases (for instances in Type 2).

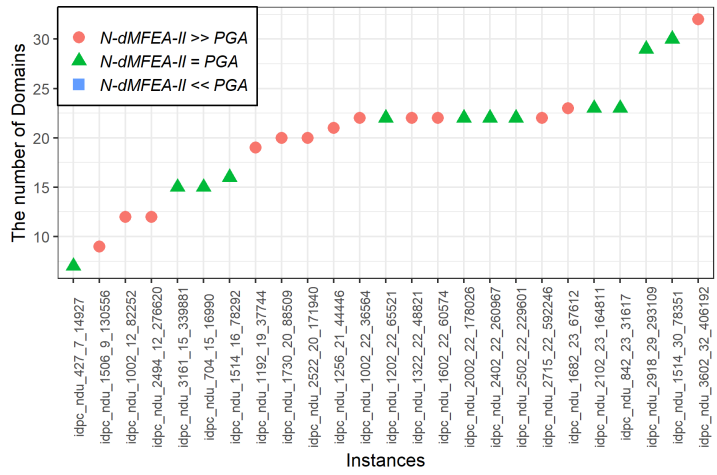
5.6 Experimental Results on the ITZD Dataset

To provide a wider perspective to the evaluation of the proposed algorithm, we also implemented it on the ITZD [28]. This dataset has been created by a project at the University of Adelaide, Australia, which collects data network topologies from around the world.

Table 10 shows the results obtained by N-dMFEA-II and the two existing algorithms on instances in the ITZD dataset. As shown in Table 10, neither TLGA nor PGA algorithms can find the solution on all test cases, whereas N-dMFEA-II finds the IDPC-NDU solution in all test cases. PGA has five instances in which the solution is not found on all runs, while TLGA has two instances. For example,



a) The comparison between N-dMFEA-II and TLGA



b) The comparison between N-dMFEA-II and PGA

Figure 9. The scatter of the number of domains and the comparison among algorithms on the instances in Types 1

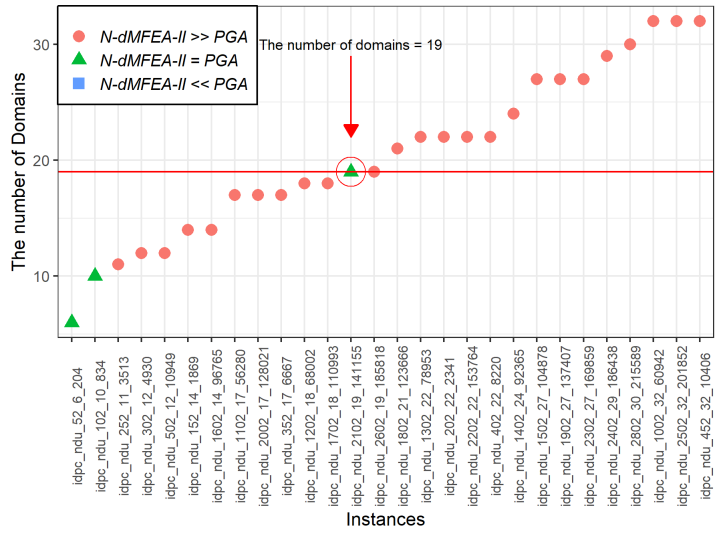


Figure 10. The scatter of the number of domains and the comparison among algorithms on the instances in Type 2

Case	Instance	N-dMFEA-II		PGA			TLGA		
		BF	Avg	NoNF	BF	Avg	NoNF	BF	Avg
1	Bandcon	130.6	130.6	–	130.6	130.6	–	130.6	130.6
	Bellcanada	105.9	105.9	–	105.9	105.9	–	105.9	105.9
2	Bics	23.4	23.4	26	***34.7***	***34.7***	–	23.4	23.4
	Claranet	5.7	5.7	–	5.7	5.7	–	5.7	5.7
3	Geant2001	5.0	5.0	–	5.0	5.0	–	5.0	5.0
	Geant2009	20.0	20.0	–	20.0	20.0	–	20.0	20.0
4	Geant2010	27.5	27.5	19	***30.0***	***30.0***	19	27.5	27.5
	Geant2012	30.0	30.0	19	30.0	30.0	–	30.0	30.0
5	HiberniaNireland	4.6	4.6	–	4.6	4.6	–	4.6	4.6
	Highwinds	4.4	4.4	–	4.4	4.4	–	4.4	4.4
6	HostwayInternational	51.8	51.8	–	51.8	51.8	–	51.8	51.8
	HurricaneElectric	123.2	123.2	–	123.2	123.2	–	123.2	123.2
7	Ntt	131.8	131.8	–	131.8	131.8	–	131.8	131.8
	Oteglobe	28.0	28.0	30	–	–	30	–	–
8	Packetexchange	92.5	92.5	–	92.5	92.5	–	92.5	92.5
	Peer1	21.6	21.6	–	21.6	21.6	–	21.6	21.6
9	Quest	331.8	331.8	–	331.8	331.8	–	331.8	331.8
	VtlWavenet2011	18.3	18.3	–	18.3	18.3	–	18.3	18.3
10	Xeex	76.7	76.7	–	76.7	76.7	–	76.7	76.7
	DeutscheTelekom	14.0	14.0	–	14.0	14.0	–	14.0	14.0
11	Gblnet	40.6	40.6	–	40.6	40.6	–	40.6	40.6

Table 10. Results obtained by TLGA, PGA and N-dMFEA-II on instances in ITZD

TLGA and PGA are unable to find the solution of the Oteglobe instance on all 30 runs; TLGA and PGA do not find the solution of the Geant2010 instance on 19 out of 30 runs.

Instance	**NVer**	**NDo**	**NEd**
Bandcon	22	6	28
Bellcanada	48	2	65
Bics	33	25	48
Claranet	15	6	18
Gblnet	8	6	7
Geant2001	27	27	38
Geant2009	34	33	52
Geant2010	37	37	58
Geant2012	40	37	61
Highwinds	18	8	53
Ntt	47	31	216
HiberniaNireland	18	2	22
HostwayInternational	16	9	21
HurricaneElectric	24	11	37
DeutscheTelekom	39	21	62
Oteglobe	93	31	106
Packetexchange	21	9	27
Peer1	16	5	20
Quest	20	12	31
VtlWavenet2011	92	7	96
Xeex	24	3	34

NoVer: The number of Vertices; **NoDo**: The number of domains; **NoEd**: The number of edges

Table 11. Summary the selected instance in the ITZD dataset

The remarkable point in the data in Table 10 is that in most cases, the results obtained by the N-dMFEA-II are equal to those obtained by the two existing algorithms. One of the reasons for this result is that the instances in the ITZD dataset have a small number of domains and vertices. Besides, Table 11 also shows that the number of links in each instance is close to the number of nodes, so the number of inter-domain paths connecting two nodes is not large. For example, Figure 11 presents the real map of the Bandcon instance, with the left side representing nodes in the US while the right side represents nodes in Europe. Since there are only 6 inter-domain edges and 6 domains, there is only a small number of inter-domain paths to consider, i.e., if the source vertex is a node in the US and the destination vertex is a node in the UK, then there are only two inter-domain paths connecting these nodes.

From the above mentioned, all three algorithms are able to find solutions and their costs are very close.

Figure 11. The real map of the Bandcon instance (source: www.topology-zoo.org)

6 CONCLUSION

This paper introduces the approach based on an adaptive multifactorial evolutionary algorithm to solve the IDPC-NDU. The proposed algorithm also describes an individual representation based on the permutation representation and a method for constructing the IDPC-NDU solution. The proposed algorithm is evaluated on a variety of different types of problem cases. The experimental results point out that the proposed algorithm performed very well in most instances and thus is very promising for future research and reference. Moreover, the relationship between the number of domains and the performance of the proposed algorithm may suggest new directions for further improvements of both the proposed algorithm and the IDPC-NDU solutions.

To enhance the performance of the proposed algorithm, in the future, the author will look for evolutionary operators and attempt to find an effective mechanism for combining the proposed algorithm with a local search algorithm.

Acknowledgment

This work is funded by the Ministry of Education and Training of Vietnam under the Contract Number B2021-TTB-01.

REFERENCES

[1] Paolucci, F.—Cugini, F.—Giorgetti, A.—Sambo, N.—Castoldi, P.: A Survey on the Path Computation Element (PCE) Architecture. IEEE Commu-

nications Surveys and Tutorials, Vol. 15, 2013, No. 4, pp. 1819–1841, doi: 10.1109/SURV.2013.011413.00087.

[2] Ogino, N.—Nakamura, H.: An Inter-Domain Path Computation Scheme Adaptive to Traffic Load in Domains. IEICE Transactions on Communications, Vol. E93-B, 2010, No. 4, pp. 907–915, doi: 10.1587/transcom.E93.B.907.

[3] Banerjee, G.—Sidhu, D.: Comparative Analysis of Path Computation Techniques for MPLS Traffic Engineering. Computer Networks, Vol. 40, 2002, No. 1, pp. 149–165, doi: 10.1016/S1389-1286(02)00270-0.

[4] Maggi, L.—Leguay, J.—Cohen, J.—Medagliani, P.: Domain Clustering for Inter-Domain Path Computation Speed-Up. Networks, Vol. 71, 2018, No. 3, pp. 252–270, doi: 10.1002/net.21800.

[5] Eiben, A. E.—Smith, J. E.: Introduction to Evolutionary Computing. 2nd Edition. Springer, 2015, doi: 10.1007/978-3-662-44874-8.

[6] Bäck, T.—Fogel, D. B.—Michalewicz, Z.: Evolutionary Computation 1: Basic Algorithms and Operators. CRC Press, 2018, doi: 10.1201/9781482268713.

[7] Bäck, T.: Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press, 1996, doi: 10.1093/oso/9780195099713.001.0001.

[8] Binh, H. T. T.—Thangy, T. B.—Long, N. B.—Hoang, N. V.—Thanh, P. D.: Multifactorial Evolutionary Algorithm for Inter-Domain Path Computation under Domain Uniqueness Constraint. 2020 IEEE Congress on Evolutionary Computation (CEC), 2020, pp. 1–8, doi: 10.1109/CEC48606.2020.9185701.

[9] Bali, K. K.—Ong, Y. S.—Gupta, A.—Tan, P. S.: Multifactorial Evolutionary Algorithm with Online Transfer Parameter Estimation: MFEA-II. IEEE Transactions on Evolutionary Computation, Vol. 24, 2020, No. 1, pp. 69–83, doi: 10.1109/TEVC.2019.2906927.

[10] Wei, T.—Wang, S.—Zhong, J.—Liu, D.—Zhang, J.: A Review on Evolutionary Multitask Optimization: Trends and Challenges. IEEE Transactions on Evolutionary Computation, Vol. 26, 2022, No. 5, pp. 941–960, doi: 10.1109/TEVC.2021.3139437.

[11] Xu, Q.—Wang, N.—Wang, L.—Li, W.—Sun, Q.: Multi-Task Optimization and Multi-Task Evolutionary Computation in the Past Five Years: A Brief Review. Mathematics, Vol. 9, 2021, No. 8, Art. No. 864, doi: 10.3390/math9080864.

[12] Osaba, E.—Del Ser, J.—Martinez, A. D.—Hussain, A.: Evolutionary Multitask Optimization: A Methodological Overview, Challenges, and Future Research Directions. Cognitive Computation, Vol. 14, 2022, No. 3, pp. 927–954, doi: 10.1007/s12559-022-10012-8.

[13] Ban, H. B.—Pham, D. H.: Multifactorial Evolutionary Algorithm for Simultaneous Solution of TSP and TRP. Computing and Informatics, Vol. 40, 2021, No. 6, pp. 1370–1397, doi: 10.31577/cai_2021_6_1370.

[14] Thanh, P. D.—Binh, H. T. T.—Trung, T. B.: An Efficient Strategy for Using Multifactorial Optimization to Solve the Clustered Shortest Path Tree Problem. Applied Intelligence, Vol. 50, 2020, No. 4, pp. 1233–1258, doi: 10.1007/s10489-019-01599-x.

[15] Osaba, E.—Martinez, A. D.—Galvez, A.—Iglesias, A.—Del Ser, J.: dMFEA-II: An Adaptive Multifactorial Evolutionary Algorithm for Permutation-Based Discrete Optimization Problems. Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion (GECCO '20), ACM, 2020, pp. 1690–1696, doi: 10.1145/3377929.3398084.

[16] Do Tuan, A.—Hoang, L. N.—Bao, T. T.—Binh, H. T. T.—Su, S.: A Two-Level Strategy Based on Evolutionary Algorithm to Solve the Inter-Domain Path Computation under Node-Defined Domain Uniqueness Constraint. In: Pham, T., Solomon, L. (Eds.): Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III. Proceedings of the SPIE, Vol. 11746, 2021, pp. 687–700, doi: 10.1117/12.2588199.

[17] Binh, H. T. T.—Long, N. H.—Thang, T. B.—Simon, S.: A Two-Level Genetic Algorithm for Inter-Domain Path Computation under Node-Defined Domain Uniqueness Constraints. 2021 IEEE Congress on Evolutionary Computation (CEC), 2021, pp. 87–94, doi: 10.1109/CEC45853.2021.9504728.

[18] Narvaez, P.—Siu, K. Y.—Tzeng, H. Y.: New Dynamic Algorithms for Shortest Path Tree Computation. IEEE/ACM Transactions on Networking, Vol. 8, 2000, No. 6, pp. 734–746, doi: 10.1109/90.893870.

[19] Chan, E. P. F.—Yang, Y.: Shortest Path Tree Computation in Dynamic Graphs. IEEE Transactions on Computers, Vol. 58, 2009, No. 4, pp. 541–557, doi: 10.1109/TC.2008.198.

[20] Gupta, A.—Ong, Y. S.—Feng, L.: Multifactorial Evolution: Toward Evolutionary Multitasking. IEEE Transactions on Evolutionary Computation, Vol. 20, 2016, No. 3, pp. 343–357, doi: 10.1109/TEVC.2015.2458037.

[21] Yuan, Y.—Ong, Y. S.—Gupta, A.—Tan, P. S.—Xu, H.: Evolutionary Multitasking in Permutation-Based Combinatorial Optimization Problems: Realization with TSP, QAP, LOP, and JSP. 2016 IEEE Region 10 Conference (TENCON), 2016, pp. 3157–3164, doi: 10.1109/TENCON.2016.7848632.

[22] Ong, Y. S.: Towards Evolutionary Multitasking: A New Paradigm in Evolutionary Computation. In: Senthilkumar, M., Ramasamy, V., Sheen, S., Veeramani, C., Bonato, A., Batten, L. (Eds.): Computational Intelligence, Cyber Security and Computational Models. Springer, Sigapore, Advances in Intelligent Systems and Computing, Vol. 412, 2016, pp. 25–26, doi: 10.1007/978-981-10-0251-9_3.

[23] Ammi, M.—Chikhi, S.: A Generalized Island Model Based on Parallel and Cooperating Metaheuristics for Effective Large Capacitated Vehicle Routing Problem Solving. Journal of Computing and Information Technology, Vol. 23, 2015, No. 2, pp. 141–155, doi: 10.2498/cit.1002465.

[24] Misevičius, A.—Ostreika, A.—Šimaitis, A.—Žilevičius, V.: Improving Local Search for the Traveling Salesman Problem. Information Technology and Control, Vol. 36, 2007, No. 2, pp. 187–195, doi: 10.5755/J01.ITC.36.2.11839.

[25] Dijkstra, E. W.: A Note on Two Problems in Connexion with Graphs. In: Apt, K. R., Hoare, T. (Eds.): Edsger Wybe Dijkstra: His Life, Work, and Legacy. ACM, 2022, pp. 287–290, doi: 10.1145/3544585.3544600.

[26] Xu, M. H.—Liu, Y. Q.—Huang, Q. L.—Zhang, Y. X.—Luan, G. F.: An Im-

proved Dijkstra's Shortest Path Algorithm for Sparse Network. Applied Mathematics and Computation, Vol. 185, 2007, No. 1, pp. 247–254, doi: 10.1016/j.amc.2006.06.094.

[27] Pham Dinh, T.—Ta Bao, T.—Hoang, N. V.—Do, T. A.: Inter-Domain Path Computation under Node-Defined Domain Uniqueness Constraint Instances. Mendeley Data, 2022, doi: 10.17632/tpg2nbcsc5.2.

[28] Knight, S.—Nguyen, H. X.—Falkner, N.—Bowden, R.—Roughan, M.: The Internet Topology Zoo. IEEE Journal on Selected Areas in Communications, Vol. 29, 2011, No. 9, pp. 1765–1775, doi: 10.1109/JSAC.2011.111002.

[29] Derrac, J.—García, S.—Molina, D.—Herrera, F.: A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms. Swarm and Evolutionary Computation, Vol. 1, 2011, No. 1, pp. 3–18, doi: 10.1016/j.swevo.2011.02.002.

[30] Carrasco, J.—García, S.—Rueda, M. M.—Das, S.—Herrera, F.: Recent Trends in the Use of Statistical Tests for Comparing Swarm and Evolutionary Computing Algorithms: Practical Guidelines and a Critical Review. Swarm and Evolutionary Computation, Vol. 54, 2020, Art. No. 100665, doi: 10.1016/j.swevo.2020.100665.

Thanh Pham Dinh received his Ph.D. degree in mathematical foundations for informatics from the Military Technical Academy in 2021. Since 2022, he has done postdoctoral research at the University of Engineering and Technology, Vietnam National University, Hanoi. Since 2006, he has joined the Faculty of Mathematics, Physics, Informatics, Tay Bac University, Vietnam as Lecturer. His current research interests include computational intelligence, evolutionary computation, memetic computing, and evolutionary multitasking. He is a member of the IEEE Computational Intelligence Society.