AMERICAN JOURNAL OF INTERDISCIPLINARY RESEARCH AND INNOVATION (AJIRI)

ISSN: 2833-2237 (ONLINE)

VOLUME 3 ISSUE 4 (2024)

4

PUBLISHED BY

E-PALLI PUBLISHERS, DELAWARE, USA

American Journal of Interdisciplinary Research and Innovation (AJIRI)

Volume 3 Issue 4, Year 2024
ISSN: 2833-2237 (Online)
DOI: https://doi.org/10.54536/ajiri.v3i4.2530
https://journals.e-palli.com/home/index.php/ajiri

An Improved Driving Training-Based Optimization Algorithm for the Minimum Gap Graph Connected Partition Problem

Ehab Morsy[1*], Riham Moharam[1]

Article Information

Received: August 16, 2024

Accepted: September 20, 2024

Published: September 23, 2024

Keywords

Connected Graph Partition, Metaheuristic Algorithms, Minimum Gap Partition, Optimization Algorithms

ABSTRACT

In this paper, we consider the minimum gap partition problem in an undirected connected graph with nonnegative vertex weights. This problem involves in dividing the given graph into a specified number of connected subgraphs such that the total difference between the largest and the smallest vertex weights in each subgraph is minimized. Namely, let $G = (V, E, W)$ be a given undirected connected graph with vertex set V , edge set E, and nonnegative vertex weight function $W : V \rightarrow R^+$, and let $k \geq 2$ be a given positive integer. The minimum gap partition problem consists of constructing a partition $P = \{V_1, V_2, \ldots, V_k\}$ of non-empty and pairwise disjoint subsets of V such that each vertex set Vi, $i = 1, 2, \ldots, k$, induces a connected subgraph $G[V_i]$ of G. The objective of the problem is to find such a partition of V that minimizes the total difference $1 \leq i \leq k$ $\max_{v \in Vi} w(v) - \min_{v \in Vi} w(v)$. This problem, as many other variants of graph partition problems, is known to be NP-hard problem. We designed an efficient metaheuristic algorithm for finding approximate solution of large-scale instances of this problem. The quality of the proposed algorithm is assessed comparing with the previous algorithms proposed for the problem.

INTRODUCTION

An interesting optimization problem is the problem of partitioning a given graph into several components that have specific characteristics on the vertices or the edges of the components that are formed. Especially in the literature, the vertex-partition problems are much more studied than the edge-partition problems. Graph partition problems have many applications in different domains such as parallel computing, community detection, image processing, wireless sensor networks, chip and circuit design, reliability of communication networks, transportation planning, cluster analysis, etc. Various forms of the graph partition problem can be found in the literature. There is a popular problem in graph theory which is the graph partition problem that is concerned with the division of a given connected graph into a prescribed number of connected subgraphs such that each subgraph has approximately the same number of vertices and edges. The weights of vertices in the graph partition problems should be such that the total weight of vertices in each subgraph is as close to one another as possible among different subgraphs. This can be achieved by minimizing the discrepancy between the total weights of various subgraphs (Chlebíková, 1996) or by adding restrictions on the total weight of all vertices in each subgraph (Ito *et al.*, 2012). For the balanced connected partition problem, we are given a vertex-weighted connected graph and the number of vertex-disjoint connected subgraphs that the graph is to be partitioned into and we are required to find a partition such that the sum of weights of all vertices in each partition is as close as possible to the average total weight of all vertices in the entire graph (Mehlhorn & Wagner, 2000). Given an edge-weighted graph, Puerto and Sainz-Pardo, 2023 studied the graph partition problem with the aim of minimizing the weight of the cut edges between the resulting components i.ethe minimum cut problem, cfalso the results in concerning the minimum cut problem (Goldschmidt & Hochbaum, 1988; Krauthgamer *et al.*, 2009). Recently, it is necessary to ensure that all vertices belonging to the same components become attached to a central vertex. In another study, Lari *et al.* (2016) gave various optimization problems having centered partitions in order to construct district maps for political elections of a specific country. In contrast, Benati *et al.* (2017) introduced various approaches to clustering rationale data.

In addition to the type of constrains and the objective functions, the graph partition problems considered in the literature differ on the type of the given graph. Most of the graph partition problems are NP-hard on general graphs. This motivates many authors to turn their attention to special classes of graphs such as trees and even special trees such as spider trees, worms, and caterpillars (De Simone *et al.*, 1990). In the special cases of partitioning trees and paths, it is possible to obtain polynomial time algorithms. The polynomial time algorithms designed for partitioning special graphs such as trees and paths have been used for several applications. They also are useful in heuristic methods for partitioning gen eral graphs (by applying them to trees or paths obtained from the given graph). The authors in (Ito *et al.*, 2012) proposed a polynomial time algorithm for partitioning a given vertex-weighted tree into subtrees with weights in a specified range. The results introduced polynomial time algorithms for partitioning trees in centered subtrees (Apollonio *et*

[1] Department of Mathematics, Suez Canal University, Ismailia 41522, Egypt
[*] Corresponding author's e-mail: ehabmorsy@science.suez.edu.eg

al., 2008; Becker *et al.*, 1998; Becker *et al.*, 1982; Bruglieri *et al.*, 2021). For path partitioning problem, Lucertini *et al.* (1993) presented a linear time algorithm that has been applied to face image degradation. Lucertini *et al.* (1993) considered the problem of partitioning a vertex-weighted path into subpaths, such that the total weight of every subpath lies in a specified range.

In this paper, we consider an important variant of the graph partition problem. Namely, given an undirected connected graph of weighted vertices, we study the k-Minimum Gap Partition problem (kMGP) for a given positive integer $k \geq 2$. The kMGP problem is an interesting NP-hard optimization problem that seeks to partition the given graph into k vertex-disjoint connected sub-graphs such that the total difference between the largest and the smallest vertex weights in each subgraph is minimized. The min-max version of the kMGP minimizes the maximum difference between the largest and the smallest vertex weights in each of the resulting subgraphs.

The kMGP problem is motivated by several real-world applications such as water distribution networks and modern ground irrigation systems. To manage a large water distribution network efficiently, we divide it into subnetworks. referred to as District Metered Areas DMAs (De Paola *et al.*, 2014). As such, if the input and the output discharges for each district area are monitored, it is possible to locate the leakages more precisely. In addition to this, we are able to minimize water losses and prevent further damages by realizing a good pressure management through pressure control valves and turbines that generate power. The optimal design of DMAs considers the objective of minimizing the difference between the heads that are needed within the DMAs. The aim is to set a unique target pressure value in every DMA, and thus, to attain an optimal pressure regulation within the network with significant ground elevation changes (Gomes *et al.*, 2015). To model the water distribution network as a graph, the edges are the pipes, the vertices are the junctions of these pipes, and the weight of the vertex is equal to the ground elevation of the junction. Therefore, the kMGP represents the search for the optimal k-partition of the water distribution network. Another potential use of the kMGP problem is the modern ground irrigation system. This system is based on the equalization of farmland as it leads to the uniformity of irrigation and distribution of the soil salt, which helps to control the weeds, save water, and raise the crop productivity (Li Xiao *et al.*, 2015). It may not be feasible to level an entire sloping land. In this regard, it is imperative to subdivide the land into plots, and then construct a flat terrace on each plot through proper earthworks. Selection of parcels with the lowest possible difference of the ground elevation allows minimizing the ground to be moved, and, therefore, the cost of the subsequent earthworks (Albornoz *et al.*, 2020). Proposes an optimization problem of partitioning an agricultural field into the minimum number of rectangular zones, with an upper bound on the variance of a suitable soil property. The kMGP problem captures the sub-division of a terrain into plots with a restricted variance in land characteristics such as elevation. The nodes of the graph represent the locations sampled in the land, the weights represent their respective heights, and the edges connect consecutive locations. Next, we present a formal description of the kMGP problem. To be self- contained, we first present the following well-known definitions.

LITERATURE REVIEW

Definition 1

Given a nonempty set V, the set $P = \{V_1, V_2, \ldots, V_k\}$ of nonempty subsets of V is called a partition of V if

1. All subsets in P are pairwise disjoint, i.e., $V_i \cap V_j = \emptyset$ for every $i, j = 1, 2, \ldots, k$, $i j$, and
2. The union of all subsets in P is V, i.e., $\cup_{i=1}^{k} Vi = V$.

Definition 2

Let $G = (V, E)$ be a given graph with vertex set V and edge set E. For a nonempty subset V' of V, the subgraph $G[V'] = (V', E')$ induced by V' is the subgraph of G such that E' is the set of all edges in E that have both endpoints in V'.

Let $G = (V, E, W)$ be a given undirected connected graph with vertex set V, edge set E, and vertex weights function $W: V \rightarrow R+$. Given a positive integer $k \geq 2$, the kMGP problem consists of constructing a partition $P = \{V_1, V_2, \ldots, V_k\}$ of the vertex set V such that each vertex set V_i, $i = 1, 2, \ldots, k$, induces a connected subgraph $G[V_i]$ of G. The objective of the kMGP problem is to find such a partition P that minimizes $\Sigma_{1 \leq i \leq k} \max_{v \in Vi} w(v) - \min_{v \in Vi} w(v)$. The kMGP problem has been introduced by Bruglieri and Cordone (2016). They proved that the problem is NP-hard and studied a couple of its special cases. Other special cases of the problem have been characterized by Bruglieri and Cordone (2021). Bruglieri *et al.* (2017) applied the basic Tabu Search heuristic procedure on small random instances of the problem. Afterwards, Bruglieri *et al.* (2017) proposed a two-level Tabu Search algorithm and an adaptive large neighborhood search algorithm to solve the problem in reasonable time on instances with up to about 23000 vertices. A Mixed Integer Linear Programming formulation of the problem is presented by Bruglieri *et al.* (2017).

Since the kMGP problem is NP-hard, it is unlikely that there is a polynomial time algorithm that finds an optimal partition of the problem in large graphs. Therefore, all practical algorithms are heuristics that differ with respect to time complexity as well as the quality of the resulting partition. Several approximation algorithms have been proposed to solve different variants and special cases of the problem in the literature.

Tang *et al.* (2014) considered a related problem to the kMGP problem. Namely, given a vertex-weighted graph, they studied the problem of partitioning the graph into districts and the total weight of each district should be the closest possible to a given value. The problem of partitioning the given graph G into clusters, each of

which contains a given special vertex (center) such that the total assignment costs of all vertices to the centers are minimized was addressed by Lari *et al.*, (2016). They also studied a variation of this problem where the objective is to minimize the sum of the costs of all the clusters. Hubert (1974) proposed the minimum diameter partitioning problem which is stated as follows given a set of objects, divide the objects into a specified number of clusters, so that the maximum dissimilarity between objects in the same cluster is minimized. This problem can be described as follows, let's consider a vertex-weighted complete graph and an instance of the kMGP problem. Especially, if the objects are assigned weights and the dissimilarity between two objects is the difference of their weights, the minimum diameter partition problem is a special case of the kMGP problem on a complete graph. This special case of complete graph is polynomially solvable (Hochbaum, 2019).

The remaining sections of the paper are arranged as below. The algorithm for the kMGP problem is given in section 2. The quality of the algorithm suggested is assessed by the analysis of the computational experiments provided in Section

3. Finally, we conclude our contributions and discuss possible future work in Section 4.

Driving Training-Based Optimization Algorithm (DTBO): An Overview

The driving training-based optimization algorithm (DTBO) is one of the most recent metaheuristic algorithms proposed by Dehghani *et al.* for dealing with the optimization problems in real-world scenarios (Dehghani, 2022). DTBO emulates the behaviour of a human being in the driving training. The driving instructor training program and the learning process for new drivers in driving schools were the main sources of inspiration for the DTBO design. Driving training offers a smart way of training a new driver to acquire the skills for driving. A novice as a student driver has a choice of instructors when registering at a driving school. The learner driver is then taught the instructions and skills by the instructor. The learner driver attempts to be taught driving skills by the instructor and drives in the tracks of the instructor. Moreover, independent practice can improve the learner's driving skills.

The population in DTBO is members with a number of including learners and instructors. The DTBO group members symbolize the individual answers to the problem at hand.Individual solutions in DTBO are updated according to the following three phases:

1. Driving instructor trains the learner driver (exploration phase). In the first stage of the DTBO update, the learner driver is presented with a driving instructor who avails himself or herself to instruct the learner driver in driving. The best population of the DTBO population is referred to as the instructors and the rest of the population is the learners. After choosing the driving instructor and acquiring the skills, members of the population will move to different places within the search area. This will increase the exploration rate of DTBO in the search space and obtain the optimal position. Therefore, this stage of the DTBO update demonstrates the search capacity of the algorithm.The following is how this DTBO phase is mathematically represented: To start with, we find out the new position for each of the members by applying equation 1. If this new position improves the value of the objective function, it replaces the old one as per Equation 2.

$$x^{P1}_{ij} = x_{i,j} + r.(DI_{ki,j} - I.x_{i,j}) \quad FDI_{ki} < F_i \qquad (1)$$
$$x_{i,j} + r.(x_{i,j} - DI_{ki,j}) \text{ otherwise}$$

$$X_i = X^{P1} \quad F^{P1}{}_i < F_i \qquad (2)$$
$$X_i \quad \text{otherwise}$$

where $X^{P1}{}_i$ is the new position for the ith individual solution on the DTBO first phase P1, X^{P1} is its jth dimension, F^{P1} is its objective value, I is a number randomly selected from the set {1, 2}, r is a random number in the interval [0, 1]. *DI* is a matrix of driving instructors, such that DI_i is the ith driving instructor, $DI_{i,j}$ is the jth dimension. The number of driving instructors is $N_{DI} = \lfloor 0.1 \cdot N \cdot (1 - t/T) \rfloor$, where t is the current iteration and T is the maximum number of iterations. DIki , where ki is randomly selected from the set {1, 2, . . . , N_{DI}}, represents a randomly selected driv ing instructor to train the ith member, $DI_{ki,j}$ is its jth dimension, and F_{DIk} is its objective value.

2. Patterning the learner driver from instructor skills (exploration phase). The learner driver imitates the instructor in the second phase of the DTBO update, trying to mimic all of the instructor's movements and driving skills. The DTBO's exploration rate is increased by this procedure, which moves members to a set of points in the search area. To mathematically model this idea, a new position is formed as a linear combination between every member and the teacher as defined by Equation.

3. If the value of the objective function is better for this new position, it shall replace the previous position as indicated by equation 4.

$$x^{P2}_{ij} = P.x_{i,j} + (1 - P).DI_{ki,j} \qquad (3)$$

$$X_i = X^{P2}{}_i \quad F^{P2}{}_i < F_i \qquad (4)$$
$$X_i \quad \text{otherwise}$$

where $X^{P2}{}_i$ is the new calculated status for the ith candidate solution based on the second phase of DTBO, $X^{P2}{}_{ij}$ is its jth dimension, $X^{P2}{}_i$ is its objective function value, and *P* is the patterning index given by Equation 5.

$$P = 0.01 + 0.9(1 - t/T) \qquad (5)$$

4. Personal practice of the learner driver (exploitation phase). The third phase of the DTBO update is based on each learner driver's individual practice of enhancing and improving their driving skills. In this phase, each learner driver aims to attain his highest level of skills. This phase allows each member to find a better position based on a local search around its current position. As a result, this phase of the DTBO update reveals the exploitation ability of the algorithm. To mathematically model this phase, a random position is first created close to each

member of the population using Equation 6. After that, if this position enhances the objective value, it replaces the prior position in accordance with Equation 7.

$$x^{p3}_{ij} = x + (1 - 2r).R.(1 - t/T).x \quad (6)$$

$$X_i = X^{p3}_i \quad F^{p3}_i < F_i \quad (7)$$
$$X_i \quad \text{otherwise}$$

where X^{p3}_i is the new calculated status for the ith candidate solution based on the third phase of DTBO, x^{p3}_{ij} is its jth dimension, X^{p3}_i is its objective function value, r is a random real number of the interval [0, 1], R is the constant set to the value 0.05, t is the counter of iterations and T is the maximum number of iterations.

3. The Proposed IDTBO Algorithm

This section introduces a new version of the DTBO algorithm for the kMGP problem. The standard DTBO was proposed to tackle continuous problems. This motivated us to propose an improved driving training-based optimization algorithm (IDTBO), which is a discrete variation of the DBTO algorithm. Each learner driver represents a feasible solution for the problem. The driving instructors represent the best solutions. To update the position of each individual solution in IDTBO, multi-point crossover, mutation and swap operators are applied to the original equations of the three DTBO phases.

The steps of the proposed IDTBO algorithm are illustrated in Algorithm 1 and are explained in more depth below.

Representation

For a specific instance of the kMGP problem, each individual solution denotes a feasible partition. The solution (partition) is represented by a binary matrix P with dimensions $k \times n$, where k is the number of subgraphs and n is the number of vertices of the graph. Each matrix element $pij \in \{0, 1\}$, i.e., $i = 1, 2, \ldots,$ k and $j = 1, 2, \ldots, n$. The element p_{ij} is set to 1 if the vertex j belongs to subgraph i and 0 otherwise. An illustrative example is shown in Figure 1.

IDTBO Initialization

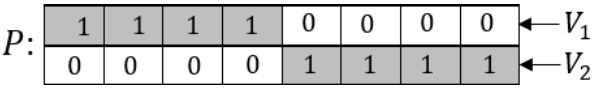

Figure 1: An example of a representation of an individual solution.

The initial population is created by applying the random initialization technique. More precisely, each vertex is assigned to a random subgraph selected from a set of k subgraphs. As long as there are n vertices, we compute each solution in the initial population by periodically using this straightforward technique. Using the well-known depth first-search algorithm, each subgraph's connectivity is tested. The resulted solution is added to the initial population only if it induces a connected subgraphs. The aforementioned process is repeated as long as the population size is less than a predefined population size pop size.

Evaluation Process

According to a fitness value, each partition (individual solution) in the population is evaluated. Recall that, the objective value of partition P is $\Sigma_{1 \le i \le k} \max_{v \in Vi} w(v) - \min_{v \in Vi} w(v)$, such that $P = \{V_1, V_2, \ldots, V_k\}$. The fitness value of partition P is defined as its objective value. The best fittest individual solution is the one with minimum fitness value.

Algorithm 1

The Proposed IDTBO Algorithm

Input: The problem size n, population size pop−size, driving instructors number Ninst, and maximum iterations number maxt.

Output: The best solution (partition with the minimum total difference between the largest and the smallest vertex weights in each subgraph).

Initialize the population at random x_i, $i = 1, 2, \ldots,$ pop−size. Evaluate the fitness value of all learner drivers (solutions) in the population. Initialize $t = 1$. while ($t \le$ maxt) do for $i = 1$ to pop−size do. Phase 1: Learning from the driving instructor (exploration). Obtain the driving instructors set using a fitness values comparison. Select random driving instructor from the set of instructors. Compute the new position for x_i learner driver using Equation 8. Update the position of x_i learner driver using Equation 9. Phase 2: Learner driver patterns the instructor's skills (exploration). Compute the new position for xi learner driver using Equation 10. Update the position of xi learner driver using Equation 11. Phase 3: Personal practice (exploitation). Compute the new position for x_i learner driver using Equation 12. Update the position of x_i learner driver using Equation 13. End for Evaluate the fitness value of all new population solutions. Update the best solutions (driving instructors). $t = t + 1$. End while return the best solution.

Updating Process

According to the standard DTBO, all individual solutions are updated according to three phases over the course of iterations. We adjusted the DTBO's original equations and arithmetic operators in the three phases to make them more suited to our problem. Multi-point crossover, mutation and swap operators are integrated into the proposed algorithm to satisfy this modification.

The following are the position updating equations for the three phases:

1. Training the learner driver by the driving instructor (exploration). Initially, a set of the population's best solutions are chosen to serve as the driving instructors. Formally, we select a set of N_{inst} driving instructors from the population with $N_{inst} = {}^{pop\_size}$, where pop_size is the population size. Each individual solution in the first phase of IDTBO is updated according to Equation 8. According to Equation 9, the new solution replaces the old one if it minimizes the value of the objective function.

$$x^{P1} = x_i \oplus r \otimes (x_{inst} - x_i) \quad (8)$$

$$x_i = x^{P1}{}_i \quad F^{P1}{}_{xi} < F_{xi} \quad (9)$$

x_i otherwise

where $x^{P1}{}_i$ is the new computed status for the x_i current solution based on the first phase of IDTBO and $F^{P1}{}_{xi}$ is its fitness value. r is a random number in the range of [0, 1] and the operator $\oplus$ is a combination operator. The operator $\otimes$ means the probability of r that all swap operators are selected from the set of swap sequence $(x_{inst} - x_i)$, where x_{inst} is a random driving instructor selected from the set of driving instructors $\{1, 2, \ldots, N_{inst}\}$.

Figure 2 illustrates an example of extracting a swap operator from $(x_{inst} - x_i)$. Suppose x_i and xinst are two partitions as seen in Figure 2. First, a random vertex is selected from the set of vertices whose positions in xi and xinst disagree. Then, the swap operator is created by its position in the two partitions, i.e., $v_l(V_i, V_j)$, where $l = 1, 2, \ldots, n$ and $i, j = 1, 2, \ldots, k$. For example, the swap operator $v_4(V_1, V_2)$, where V_1 is the position of v_4 in xi and V_2 in xinst, means v_4 can be swapped from V_1 to V_2. Finally, xi is updated using the selected swap operator as shown in Figure 2. After every update, the depth first search algorithm is applied to verify that subgraphs are connected.

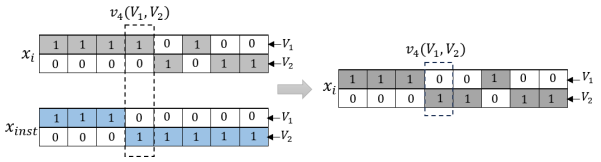

Figure 2: An example of generating a swap operator between two solutions.

2. Patterning the learner driver from instructor skills (exploration). The learner driver uses the crossover process to mimic all of the instructor's moves and skills during the second IDTBO phase. Each learner driver (individual solution) in this phase of IDTBO is updated according to Equation 10 and 11.

$$x^{P2}{}_i = x_i \boxplus x_{inst} \quad (10)$$

$$x_i = x^{P2}{}_i \quad F^{P2}{}_{xi} < F_{xi} \quad (11)$$

x_i otherwise

where $x^{P2}{}_i$ is the new computed status for the x_i solution based on the second phase of IDTBO and $F^{P2}{}_{xi}$ is its fitness value. The operator $\boxplus$ is a crossover operator, it means applying crossover between x_i and x_{inst}. Figure 3 illustrates an example of crossover process. A multipoint crossover operator is applied with m random points selected from the two partitions x_i and x_{inst} with m = log n, where n is the number of vertices. The segments between m points are then swapped between x_i and x_{inst} to get a new partition. We consider the new generated partition the new status of x_i. To increase the exploration rate, we apply the crossover process with a probability of 0.9.

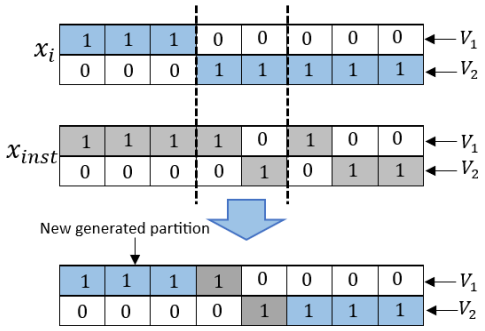

Figure 3: An example of crossover process.

3. Personal practice of the learner driver (exploitation). During this phase of IDTBO, each learner driver applies the mutation process in an attempt to get closer to his best skills. The position of each learner driver is updated according to Equation 12 and 13.

$$x^{P3} = R \boxdot x_i \quad (12)$$

$$x_i = x^{P3}{}_i \quad F^{P3}{}_{xi} < F_{xi} \quad (13)$$

x_i otherwise

where $x^{P3}{}_i$ is the new computed status for the x_i solution based on the third phase of IDTBO and $F^{P3}{}_{xi}$ is its fitness value. The operator $\boxdot$ represents a mutation operator. A swap mutation operator is applied on xi with a probability R. In order to fulfill the exploitation process, we set R = 0.2. Figure 4 illustrates an example of mutation process.

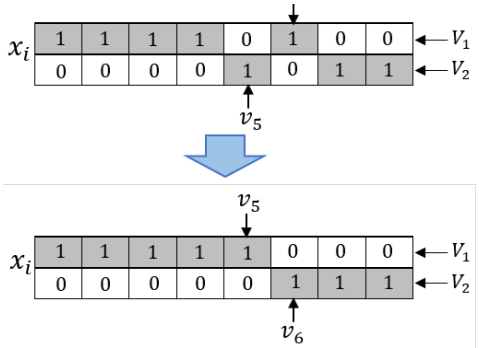

Figure 4: An example of mutation process.

Last Steps of IDTBO

The proposed IDTBO selects the best solution (partition with the minimum total difference between the largest and the smallest vertex weights in each subgraph) after a fixed number of iterations.

METHODOLOGY

To prove the efficiency of IDTBO, we test it on some random instances of the kMGP problem. In addition, the performance of IDTBO is benchmarked with that of other meta-heuristic algorithms, GA (Engelbrecht, 2007), PSO (Kennedy & Eberhart, 1995), GWO (Mirjalili *et al.*, 2014), and ChOA algorithm (Khishe & Mosavi., 2020). All the algorithms were implemented in MATLAB R2016a, and the machine used for the running the MATLAB program has Windows 10 64-bit operating

system, Intel Core i7-7500U processor having 2.70 GHz, and 16 GB of RAM.

RESULTS AND DISCUSSION

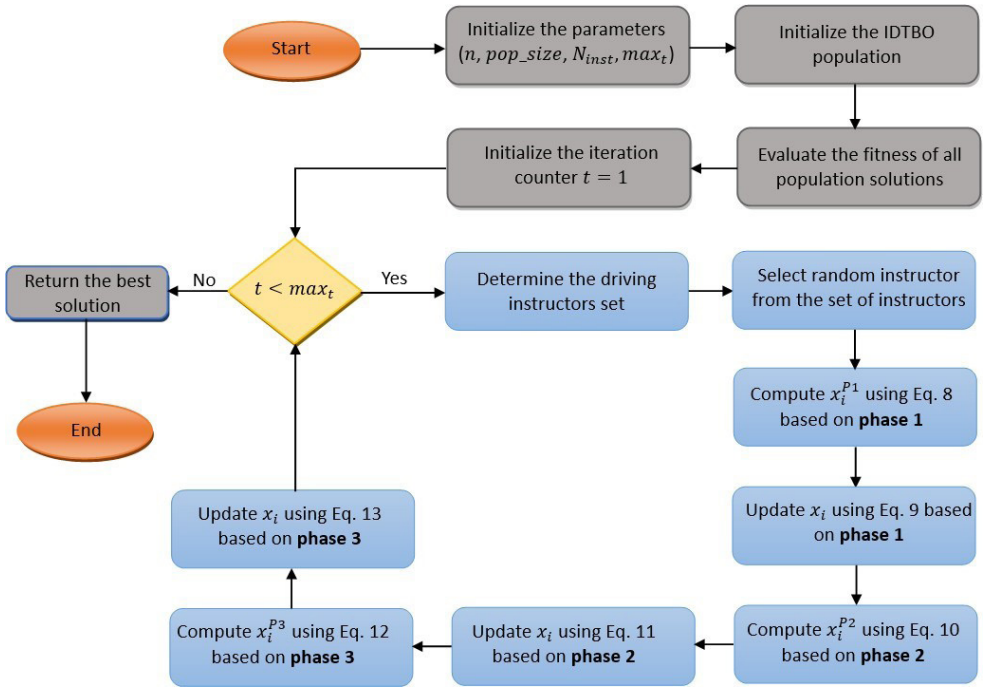

Figure 5: Flowchart of the proposed IDTBO.

Parameter Configuration

The IDTBO algorithm is evaluated on random instances generated by the Erdos and Renyi method (Erdos & Renyi, 1959). Graphs in the sizes of 50, 100, 150, 200, and 250 are generated at random, and the vertex weights are chosen from a uniform distribution in the range of [1, 100]. The graphs are partitioned into k subgraphs with values 2, 3, 4, and 5.

The population size and maximum number of iterations values changed according to the graph size. Based on experiments, the population size (pop size) is set to n/5 and the maximum number of iterations (maxt) is set to n/2, where n is the number of vertices of the graph. Each algorithm performs 20 independent runs for each instance size. Table 1 list all parameters and their values.

Table 1: Parameter configuration for IDTBO

Parameter	Value
Graph sizes (n)	50, 100, 150, 200, 250
Number of subgraphs (k)	2, 3, 4, 5
Population size (pop size)	n/5
Number of iterations (maxt)	n/2
Number of independent runs	20

Results and Analysis

The first experiment compares the IDTBO's fitness values to those of other well-known metaheuristic algorithms (GA, PSO, GWO, ChOA). The comparison in Table 2 is according to the best and worst fitness values over 20 runs for each graph size. Bold values denotes the best values. Table 2 demonstrates that IDTBO gives superior results in terms of the best and worst fitness over the majority of instances. The average fitness values for all algorithms over 20 runs for each graph size are depicted in Figure 6. The figure shows that the IDTBO has attained the minimum average fitness value in most cases.

The convergence of the proposed IDTBO in comparison to other metaheuris tic algorithms can be seen for all graph sizes in Figure 7. According to the figure's observation, IDTBO outperformed all other algorithms

Table 2: Fitness values of the IDTBO and other metaheuristics over all instances.

Algorithm	Fitness	n = 50	n = 100	n = 150	n = 200	n = 250
GA	Best	33	81	128	160	328
	Worst	36	85	142	166	337
PSO	Best	44	88	126	166	310
	Worst	53	92	127	170	312
GWO	Best	35	85	107	137	307
	Worst	37	90	110	140	311
ChOA	Best	31	82	105	134	298
	Worst	35	85	111	137	300
IDTBO	Best	25	74	107	119	201
	Worst	26	89	109	125	205

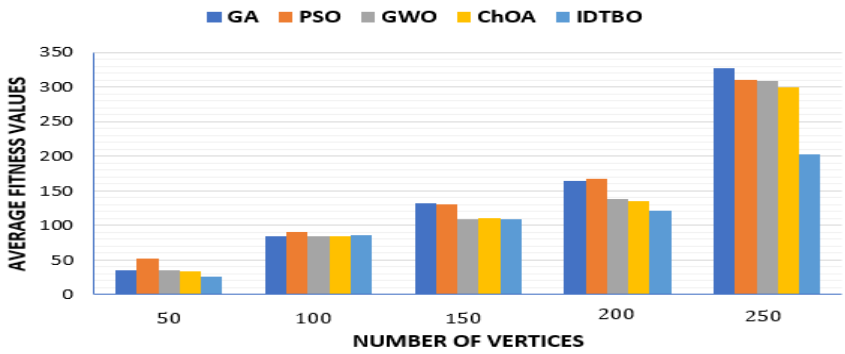

Figure 6: Total average fitness of IDTBO compared with other algorithms over all instances.

and demonstrated superior convergence performance over the majority of the graph sizes. Hence, we can conclude that the IDTBO avoided early convergence in most graph sizes by achieving a balance between exploration and exploitation throughout the course of the three phases.

Another experiment is carried out to compare the IDTBO with other algorithms according to the average and maximum relative errors between the solutions of the algorithms and the optimal ones. Optimal solutions determined by taking into account all possibilities of feasible partitions in the underlying graphs. The results of all algorithms are displayed in Table 3 based on the average and maximum relative errors over 20 runs for each graph size. The following is the relative error calculation formula:

$$Rl_{er} = (F_{alg} - F_{opt}) / F_{opt} \times 100\ \% \quad (14)$$

where F_{alg} stands for the fitness values obtained by the proposed algorithms. The optimal fitness value is represented by F_{opt}. The best results are highlighted in

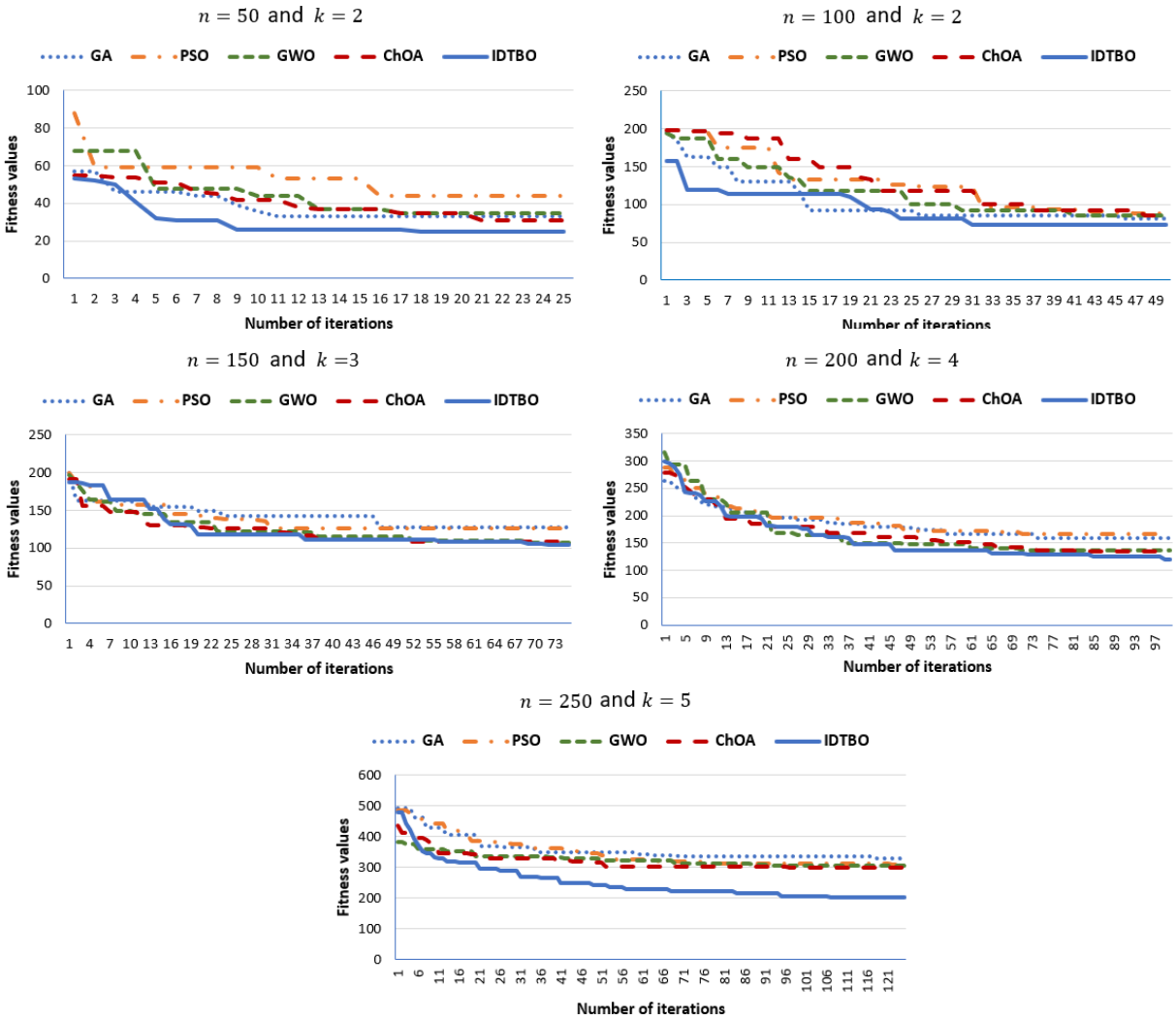

Figure 7: The convergence of proposed IDTBO compared with other metaheuristics.

bold font in Table 3. Observation of this table reveals that IDTBO has a higher performance than other metaheuristic algorithms. In particular, IDTBO has the minimum values for the average and maximum relative errors for most graph sizes. Table 3 leads us to the conclusion that the IDTBO can obtain near optimal solutions for a majority of graph sizes. Figure 8 depicts the average running time in seconds for all algorithms over 20 runs for each graph size.

Table 3: Comparison between IDTBO and other metaheuristics according to the average and maximum relative error.

n	GA		PSO		GWO		ChOA		IDTBO	
	avg_{er}	max_{er}	avg_{er}	max_{er}	avg_{er}	max_{er}	avg_{er}	max_{er}	avg_{er}	max_{er}
50	0.184	0.196	0.237	0.362	0.147	0.189	0.135	0.168	0.117	0.151
100	0.191	0.207	0.285	0.394	0.175	0.195	0.166	0.185	0.133	0.148
150	0.232	0.286	0.373	0.495	0.211	0.228	0.27	0.289	0.193	0.215
200	0.377	0.46	0.518	0.608	0.322	0.39	0.302	0.318	0.281	0.325
250	0.53	0.566	0.637	0.682	0.421	0.436	0.337	0.389	0.351	0.375

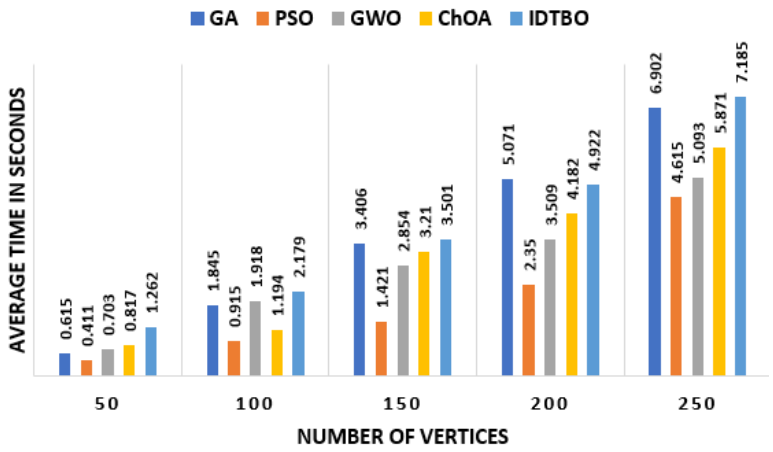

Figure 8: Average running time of IDTBO compared with other algorithms over all instances.

CONCLUSION

We have studied the k Minimum Gap Partition problem (kMGP) in an undirected connected graph with nonnegative vertex weights. The problem involves in partitioning the given graph into k connected subgraphs such that the total difference between the largest and the smallest vertex weights in the resulting subgraphs is minimized. The kMGP problem is known to be NP-hard. It has several real-world applications that motivate the importance of designing and implementing algorithms that find near optimal partitions in reasonable time.

We have designed an improved driving training-based optimization algorithm (IDTBO) for the kMGP problem. In IDTBO, variables are permuted using a swap operator. Moreover, multi-point crossover and mutation operators are integrated in IDTBO. The proposed IDTBO is compared to various well-known metaheuristic algorithms (GA, PSO, GWO, and ChOA) in order to verify its effectiveness and assess its performance. According to the findings of experiments, IDTBO performed better than other algorithms for most instances. Future research could focus on studying various problem variations with various objective functions and constraints. Developing and implementing novel, more effective algorithms for this problem will be interesting as well.

REFERENCES

Albornoz, V. M., Véliz, M. I., Ortega, R., & Ortíz-Araya, V. (2020). Integrated versus hierarchical approach for zone delineation and crop planning under uncertainty. *Annals of Operations Research, 286*, 617-634.

Apollonio, N., Lari, I., Ricca, F., Simeone, B., & Puerto, J. (2008). Polynomial algorithms for partitioning a tree into single-center subtrees to minimize flat service costs. *Networks: An International Journal, 51*(1), 78-89.

Becker, R., Lari, I., Lucertini, M., & Simeone, B. (1998). Max-min partitioning of grid graphs into connected components. *Networks: An International Journal, 32*(2), 115-125.

Becker, R. I., Schach, S. R., & Perl, Y. (1982). A shifting algorithm for min-max tree partitioning. *Journal of the ACM (JACM), 29*(1), 58-67.

Benati, S., Puerto, J., & Rodriguez-Chia, A. M. (2017). Clustering data that are graph connected. *European Journal of Operational Research, 261*(1), 43-53.

Bruglieri, M., & Cordone, R. (2016). Partitioning a graph into minimum gap components. *Electronic Notes in Discrete Mathematics, 55*, 33-36.

Bruglieri, M., Cordone, R., & Caurio, V. (2017, June). A metaheuristic for the minimum gap graph partitioning problem. In *Proceedings of the 15th Cologne-Twente Workshop on Graphs and Combinatorial Optimization* (pp.

23-26).

Bruglieri, M., & Cordone, R. (2021). Metaheuristics for the minimum gap graph partitioning problem. *Computers & Operations Research, 132*, 105301.

Bruglieri, M., Cordone, R., Lari, I., Ricca, F., & Scozzari, A. (2021). On finding connected balanced partitions of trees. *Discrete Applied Mathematics, 299*, 1-16.

Chlebíková, J. (1996). Approximating the maximally balanced connected partition problem in graphs. *Information Processing Letters, 60*(5), 225-230.

De Paola, F., Fontana, N., Galdiero, E., Giugni, M., degli Uberti, G. S., & Vitaletti, M. (2014). Optimal design of district metered areas in water distribution networks. *Procedia Engineering, 70*, 449-457.

De Simone, C., Lucertini, M., Pallottino, S., & Simeone, B. (1990). Fair dissections of spiders, worms, and caterpillars. *Networks, 20*(3), 323-344.

Dehghani, M., Trojovská, E., & Trojovský, P. (2022). A new human-based metaheuristic algorithm for solving optimization problems on the base of simulation of driving training process. *Scientific reports, 12*(1), 9924.

Engelbrecht, A. P. (2007). *Computational intelligence: an introduction*. John Wiley & Sons.

Erdos, P. & Renyi, A. (1959). On random graphs. *Publ. Math 6*(18), 290-297.

Goldschmidt, O., & Hochbaum, D. S. (1988, October). Polynomial algorithm for the k-cut problem. In [Proceedings 1988] *29th Annual Symposium on Foundations of Computer Science* (pp. 444-451). IEEE Computer Society.

Gomes, R., Sousa, J., Muranho, J., & Marques, A. S. (2015). Different design criteria for district metered areas in water distribution networks. *Procedia Engineering, 119*, 1221-1230.

Hochbaum, D. S. (2019). Algorithms and complexity of range clustering. *Networks, 73*(2), 170-186.

Hubert, L. J. (1974). Some applications of graph theory to clustering. *Psychometrika, 39*(3), 283-309.

Ito, T., Nishizeki, T., Schröder, M., Uno, T., & Zhou, X. (2012). Partitioning a weighted tree into subtrees with weights in a given range. *Algorithmica, 62*, 823-841.

Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks* (Vol. 4, pp. 1942-1948). ieee.

Khishe, M., & Mosavi, M. R. (2020). Chimp optimization algorithm. E*xpert systems with applications, 149,* 113338.

Krauthgamer, R., Naor, J., & Schwartz, R. (2009, January). Partitioning graphs into balanced components. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms* (pp. 942-949). Society for Industrial and Applied Mathematics.

Lari, I., Ricca, F., Puerto, J., & Scozzari, A. (2016). Partitioning a graph into connected components with fixed centers and optimizing cost-based objective functions or equipartition criteria. *Networks, 67*(1), 69-81.

Li Xiao, L. X., Li HongPeng, L. H., Niu DongLing, N. D., Wang Yan, W. Y., & Liu Gang, L. G. (2015). Optimization of GNSS-controlled land leveling system and related experiments, *31*(3), 48–55.

Lucertini, M., Perl, Y., & Simeone, B. (1993). Most uniform path partitioning and its use in image processing. *Discrete Applied Mathematics, 42*(2-3), 227-256.

Mehlhorn, K. & Wagner, U. (2000). Connected k-partition problems. *Journal of Al- gorithms, 37*(1), 1-28.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in engineering software, 69*, 46-61.

Puerto, J., & Sainz-Pardo, J. L. (2023). *Partitioning a graph constraining the weight of its cuts*. Available at SSRN 4341526.

Tang, X., Soukhal, A., & T'kindt, V. (2014). Preprocessing for a map sectorization problem by means of mathematical programming. *Annals of Operations Research, 222,* 551-569.