# An iterated local search algorithm for the minimum differential dispersion problem

Yangming Zhou [a] and Jin-Kao Hao [a,b,*]

[a] *LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France*

[b] *Institut Universitaire de France, 1 rue Descartes, 75231 Paris, France*

***Knowledge-Based Systems, Doi: 10.1016/j.knosys.2017.03.028***

**Abstract**

Given a set of $n$ elements separated by a pairwise distance matrix, the minimum differential dispersion problem (*Min-Diff DP*) aims to identify a subset of $m$ elements ($m < n$) such that the difference between the maximum sum and the minimum sum of the inter-element distances between any two chosen elements is minimized. We propose an effective iterated local search (denoted by ILS_MinDiff) for *Min-Diff DP*. To ensure an effective exploration and exploitation of the search space, ILS_MinDiff iterates through three sequential search phases: a fast descent-based neighborhood search phase to find a local optimum from a given starting solution, a local optima exploring phase to visit nearby high-quality solutions around a given local optimum, and a local optima escaping phase to move away from the current search region. Experimental results on six data sets of 190 benchmark instances demonstrate that ILS_MinDiff competes favorably with the state-of-the-art algorithms by finding 131 improved best results (new upper bounds).

*keywords*: Combinatorial optimization; dispersion problems; heuristics; iterated local search; three phase search.

## 1 Introduction

Let $N = \{e_1, e_2, \ldots, e_n\}$ be a set of $n$ elements and $d_{ij}$ be the distance between $e_i$ and $e_j$ according to a given distance metric such that $d_{ij} > 0$ if $i \neq j$ and $d_{ij} = 0$ otherwise. The minimum differential dispersion problem (*Min-Diff DP*) is to identify a subset $S \subset N$ of a given cardinality $m$ ($m < n$), such

* Corresponding author.

*Email addresses:* zhou.yangming@yahoo.com (Yangming Zhou), hao@info.univ-angers.fr (Jin-Kao Hao).

that the difference between the maximum sum and the minimum sum of the inter-element distances between any two elements in $S$ is minimized. Formally, *Min-Diff DP* can be described in the following way.

Let $\Delta(e_v)$ be the sum of pairwise distances between an element $e_v \in S$ and the remaining elements in $S$, that is:

$$\Delta(e_v) = \sum_{e_u \in S, u \neq v} d_{uv} \tag{1}$$

The objective value $f$ of the solution $S$ is then defined by the following differential dispersion:

$$f(S) = \max_{e_u \in S}\{\Delta(e_u)\} - \min_{e_v \in S}\{\Delta(e_v)\} \tag{2}$$

Then, *Min-Diff DP* is to find a subset $S^* \subset N$ of size $m$ with the minimum differential dispersion, i.e.,

$$S^* = \arg\min_{S \in \Omega} f(S) \tag{3}$$

where $\Omega$ is the search space including all possible subsets of size $m$ in $N$, i.e., $\Omega = \{S : S \subset N \text{ and } |S| = m\}$. The size of $\Omega$ is extremely large, up to a maximum number of $\binom{n}{m} = \frac{n!}{m!(n-m)!}$.

*Min-Diff DP* is one of many diversity or dispersion problems [1,30] which basically aim to find a subset $S$ from a given set of elements, such that a distance-based objective function over the elements in $S$ is maximized or minimized. These problems can be further classified according to two types of objective functions:

- *Efficiency-based* measures which consider some dispersion quantity for all elements in $S$. This category mainly includes the *maximum diversity problem* (MDP) and the *max-min diversity problem* (MMDP), which respectively maximizes the total sum of the inter-element distances of any two chosen elements and the minimum distance of any two chosen elements.
- *Equity-based* measures which guarantee equitable dispersion among the selected elements. This category includes three problems: (i) the *maximum mean dispersion problem* (Max-Mean DP) maximizes the average inter-element distance among the chosen elements; (ii) the *maximum min-sum dispersion problem* (Max-Min-sum DP) maximizes the minimum sum of the inter-element distances between any two chosen elements; (iii) the *minimum differential dispersion problem* considered in this work. It is worth

noting that the cardinality of subset $S$ is fixed except for Max-Mean DP.

In addition to their theoretical significance as NP-hard problems, diversity problems have a variety of real-world applications in facility location [21], pollution control [9], maximally diverse/similar group selection (e.g., biological diversity, admission policy formulation, committee formation, curriculum design, market planning) [17,27,28], densest subgraph identification [20], selection of homogeneous groups [8], web pages ranking [19,33], community mining [32], and network flow problems [7].

In this study, we focus on *Min-Diff DP*, which is known to be strongly NP-hard [30]. *Min-Diff DP* can be formulated as a 0-1 mixed integer programming problem (MIP) [30]. Thus it can be conveniently solved by MIP solvers like IBM ILOG CPLEX Optimizer (CPLEX). However, being an exact solver, CPLEX is only able to solve instances of small size (up to $n = 50$ and $m = 15$), while requiring high CPU times (more than 2500 seconds) [30]. To handle larger instances, heuristic and meta-heuristic algorithms are often preferred to find near-optimal solutions. In recent years, several heuristic approaches have been proposed in the literature [30,13,28]. In particular, in 2015, based on greedy randomized adaptive search procedure (GRASP), variable neighborhood search (VNS) and exterior path relinking (EPR), Duarte et al. proposed several effective hybrid heuristics [13]. Very recently (2016), Mladenović et al. proposed an improved VNS algorithm which uses the swap neighborhood both in its descent and shaking phases [28]. This new VNS algorithm significantly outperforms the previous best heuristics reported in [13] and is the current best performing algorithm available in the literature for *Min-Diff DP*. We will use it as our main reference for the computational studies.

Our literature review showed that, contrary to other diversity problems like MDP and Max-Min DP, for which many exact and heuristic methods have been investigated, there are only a very limited number of studies for *Min-Diff DP*. In this work, we partially fill the gap in terms of heuristic solving of *Min-Diff DP*, by proposing an effective heuristic algorithm for the problem (named ILS_MinDiff). We identify the main contributions of this work as follows.

First, in terms of algorithmic design, the proposed ILS_MinDiff algorithm is the first adaptation of the general three-phase search method to *Min-Diff DP* (Section 2). Compared to other metaheuristic framework, this method has the main advantage of being conceptually simple and easy implementation. The proposed algorithm iterates through a descent-based local optimization phase, a local optima exploring phase, and a local optima escaping phase (Section 3). Like [13,28], the descent-based local optimization phase is based on the conventional swap neighborhood. Meanwhile, ILS_MinDiff distinguishes itself from the existing heuristic algorithms by at least two specific features. Its local optima exploring phase applies a weak perturbation technique (based on

deterministic tournament selection) to discover nearby local optima around a given local optimum (Section 3.4). On the other hand, the local optima escaping phase calls for a strong perturbation technique (based on parametric random selection) to move away from deep local optimum traps (Section 3.5).

Second, in terms of performance, we present 131 improved best-known results (i.e., new lower bounds) out of 190 popular *Min-Diff DP* benchmarks ($\approx 69\%$). These improved lower bounds constitute a valuable contribution to *Min-Diff DP* studies since they can be used as new reference values to assess other *Min-Diff DP* heuristic algorithms. These tightened lower bounds could also be used within an exact algorithm for better bounding and possibly leading to an optimality proof of some benchmark instances. Together with the equal best-known results for other 42 instances, the computational outcomes confirm the relevance of the proposed algorithm.

Third, the availability of the source code of our ILS_MinDiff algorithm (see Section 4.2) contributes favorably to future research on *Min-Diff DP* and related problems. Specifically, the code can be used to perform comparative studies or solve other problems that can be formulated as Min-Diff DP. It can also serve as a key component (local optimizer) of more sophisticated *Min-Diff DP* algorithms.

The remainder of the paper is organized as follows. In the next section, we present a brief literature review on the iterated local search framework and its two recent variants. In Section 3, we describe the general framework and the key components of the proposed algorithm. In Section 4, we show an extensive experimental comparison between our algorithm and state-of-the-art algorithms. A parameter analysis is proviy45(e)-0.126613(c)613(t)33376.192(p)-(s)-0.1940033

to generate a new starting point for the next iteration (Algorithm 1).

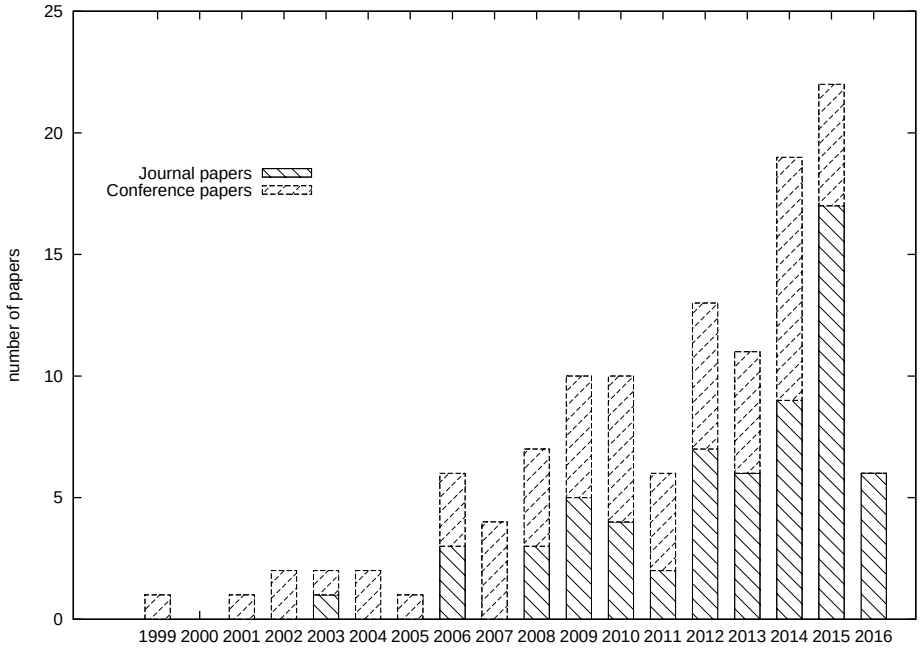


Fig. 1. Journal and conference publications by years on ILS. Data was extracted from the DBLP database http://dblp.uni-trier.de/search/publ under the keywords "iterated local search" on 29 June 2016.

**Algorithm 1** Iterated local search

1: $S_0 \leftarrow$ GenerateInitialSolution()
2: $S^* \leftarrow$ LocalSearch($S_0$)
3: **while** a stopping condition is not reached **do**
4: $S' \leftarrow$ Perturbation($S^*, history$)
5: $S^{*\prime} \leftarrow$ LocalSearch($S'$)
6: $S^* \leftarrow$ AcceptanceCriterion($S^*, S^{*\prime}, history$)
7: **end while**

Based on the general ILS framework, several variants and extended approaches have recently been proposed, including two representative methods called breakout local search (BLS) [2,6] and three-phase search (TPS) [15]. The effectiveness of BLS and TPS have been verified on a variety of hard optimization problems and applications (see examples of Table 1). In the following, we present a brief review of these ILS variants.

*Breakout local search* introduced in [2,6] combines local search with a *dedicated and adaptive perturbation* mechanism. Its basic idea is to use a descent-based local search procedure to intensify the search in a given search region, and to perform dedicated perturbations to jump into a new promising search region once a local optimum is encountered. BLS is characterized by its adaptive perturbation. At the perturbation phase, BLS attempts to achieve the most suitable degree of diversification by dynamically determining the number of

Table 1
A summary of the applications of breakout local search and three-phase search.

| Breakout local search | Three-phase search |
|---|---|
| minimum sum coloring problem [2] | quadratic minimum spanning tree problem [15] |
| quadratic assignment problem [3] | maximally diverse grouping problem [22] |
| maximum clique problem [4] | capacitated clustering problem [23] |
| max-cut problem [5] | max-k-cut problem [25] |
| vertex separator problem [6] | clique partitioning problem [34] |
| Steiner tree problem [14] | minimum differential dispersion problem |
| assembly sequence planning problem [16] | |
| single-machine total weighted tardiness problem [11] | |

perturbation moves (i.e., the jump magnitude) and by adaptively selecting between several types of pre-defined perturbation operations of different intensities, which is achieved through the use of information from specific memory structures. As summarized in Table 1, BLS has reported excellent performances for several well-known combinatorial optimization problems. Algorithm 2 describes the general framework of BLS. BLS distinguishes itself from the conventional ILS approach by the following two aspects. First, multiple types of perturbations are used in BLS, which are triggered according to the search states, achieving variable levels of diversification. Second, the local optimal solution returned by the local search procedure is always accepted as the new starting solution in BLS regardless of its quality, which completely eliminates the acceptance criterion component of ILS (Algorithm 1, line 6).

**Algorithm 2** Breakout local search

1: $S \leftarrow$ GenerateInitialSolution()
2: $L \leftarrow L_0$
3: **while** a stopping condition is not reached **do**
4: $\quad S' \leftarrow$ DescentBasedSearch($S$)
5: $\quad L \leftarrow$ DetermineJumpMagnitude($L, S', history$)
6: $\quad T \leftarrow$ DeterminePerturbationType($S', history$)
7: $\quad S \leftarrow$ Perturb($L, T, S', history$)
8: **end while**

*Three-phase search* proposed in [15] follows and generalizes the basic ILS scheme. TPS iterates through three distinctive and sequential search phases. The basic idea of TPS is described as follows. Starting from an initial solution, a descent-based neighborhood search procedure is first employed to find a local optimal solution. Then, a local optima exploring phase is triggered with the purpose of discovering nearby local optima of better quality. When the search stagnates in the current search zone, TPS turns into a diversified perturbation phase, which strongly modifies the current solution to jump into a new search region. The process iteratively runs the above three phases until a given stopping condition is met. Compared to BLS, TPS further divides the perturbation phase into a local optima exploring phase (to discover more local

optima within a given region) and a diversified perturbation phase (to displace the search to a new and distant search region). TPS has been successfully used to solve several optimization problems, as shown in the right column of Table 1. The general framework of TPS is outlined in Algorithm 3. Actually, the ILS_MinDiff algorithm proposed in this work follows the TPS framework.

**Algorithm 3** Three-phase search

```
1: S ← GenerateInitialSolution()
2: S* ← S
3: while a stopping condition is not reached do
4:     S ← DescentBasedSearch(S)
5:     S ← LocalOptimaExploring(S, history)
6:     S* ← BestOne(S, S*)
7:     S ← DiversifiedPerturb(S, history)
8: end while
```

## 3 An iterated local search algorithm for Min-Diff DP

### *3.1 General framework*

Given a set of elements $N = \{e_1, e_2, \ldots, e_n\}$, any subset set $S \subset N$ of $m$ elements is a legal or feasible solution of *Min-Diff DP* and can be represented by $S = \{e_{S(1)}, e_{S(2)}, \ldots, e_{S(m)}\}$ $(1 \leq S(i) \neq S(j) \leq n$ for all $i \neq j)$ where $S(l)$ $(1 \leq l \leq m)$ is the index of each selected element in $S$.

Following the three-phase search framework [15], the proposed ILS_MinDiff algorithm is composed of three main components: a descent-based neighborhood search procedure, a local optima exploring procedure and a local optima escaping procedure. Starting from a (good) initial solution provided by an initial solution generation procedure (Section 3.2), ILS_MinDiff first employs the descent neighborhood search procedure to quickly attain a local optimal solution (Section 3.3). Then it switches to the local optima exploring procedure which attempts to discover better solutions around the attained local optimum (Section 3.4). When no improved solution can be found (the search is stagnating in the current search zone), ILS_MinDiff tries to jump to a new region with the help of a strong perturbation operation (Section 3.5). The best solution encountered during the search is recorded in $S^b$ and updated whenever it is needed. ILS_MinDiff repeats the above three phases until a stopping condition (in our case, a time limit $t_{max}$) is reached (Algorithm 4). The composing procedures of ILS_MinDiff are presented in the next subsections.

**Algorithm 4** Framework of the ILS_MinDiff algorithm for *Min-Diff DP*

**Input**: a problem instance and the time limit $t_{max}$
**Output**: the best solution $S^b$ found
$S \leftarrow$ generate_initial_solution() /* generate a good initial solution, Section 3.2 */
$S^b \leftarrow S$ /* record the best solution found so far in $S^b$ */
**while** a stopping condition is not reached **do**
  $S \leftarrow$ descent_based_neighborhood_search($S$) /* local search, Section 3.3 */
  $S \leftarrow$ local_optima_exploring($S$) /* explore nearby local optima, Section 3.4 */
  // update the best solution found so far
  **if** $f(S) < f(S^b)$ **then**
    $S^b \leftarrow S$
  **end if**
  $S \leftarrow$ local_optima_escaping($S$) /* escape from local optima, Section 3.5 */
**end while**

### *3.2 Initialization*

The ILS_MinDiff algorithm requires an initial solution to start its search. In general, the initial solution can be generated by any means (e.g., a random procedure or a greedy heuristic). In this work, the search starts from an elite solution of good quality, which is obtained in the following way. From a random solution $S \in \Omega$ (i.e., any subset of $m$ elements), we apply the descent-based neighborhood search procedure (Section 3.3) to improve $S$ until a local optimum is reached. We repeat the process ten times to obtain ten local optimal solutions among which we select the best one (i.e., having the smallest objective value) as the initial solution. This procedure allows us to obtain an initial solution of relatively high quality. However, for large instances with $n \geqslant 3000$, this initialization process becomes too time-consuming. In this case, the algorithm starts its search with a random solution, instead of an elite solution.

### *3.3 Descent-based neighborhood search phase*

To obtain a local optimum from a given starting solution, a neighborhood search procedure is needed. In our case, we employ a simple and fast *descent_based_neighborhood_search()* procedure. This search procedure iteratively makes transitions from the incumbent solution to a new neighbor solution according to a given neighborhood relation such that each transition leads necessarily to a better solution. This improvement process runs until no improving neighbor solution is available in the neighborhood, in which case the current solution corresponds to a local optimum with respect to the neighborhood.

For the design of such a search procedure, two important issues must be con-

sidered: the neighborhood structure and the technique for a fast evaluation of neighboring solutions. The neighborhood *Neighbor* explored by the *descent_based_neighborhood_search()* procedure is based on the *swap* operation, which has been used in previous studies [13,28]. Given a solution $S$, we define $swap(p, q)$ as the move that exchanges an element $e_p \in S$ with an element $e_q \in N \setminus S$. Each $swap(p, q)$ brings about a variation $\Delta f(S, p, q)$ in the objective function $f$. Let $S'$ be the neighbor solution obtained by applying $swap(p, q)$ to the solution $S$, then the objective variation $\Delta f(S, p, q)$ (also called the move gain) is given by $\Delta f(S, p, q) = f(S') - f(S)$. Obviously, the size of this swap-based neighborhood is bound by $\mathcal{O}(m(n - m))$.

To evaluate the neighborhood as fast as possible, we adopt a popular incremental evaluation technique [13,22,28] to streamline the calculation of $\Delta f(S, p, q)$. Once a $swap(p, q)$ move is performed, only the elements related to $e_p$ and $e_q$ are needed to be considered. Before calculating $\Delta f(S, p, q)$ caused by $swap(p, q)$, we first estimate the $\Delta$ value of each element $e_w$ in $S$ as follows:

$$\Delta'(e_w) = \begin{cases} \Delta(e_w) - d_{wp} + d_{wq} & \forall e_w \in S \setminus \{e_p\} \\ \sum_{e_z \in S \setminus \{e_p\}} d_{qz} & e_w = e_q \end{cases} \tag{4}$$

Therefore, with the $swap(p, q)$ operation, the objective value of the resulting neighbor solution $S' = S \setminus \{e_p\} \cup \{e_q\}$ can be conveniently calculated according to the following formula:

$$f(S') = \max_{e_w \in S'} \{\Delta'(e_w)\} - \min_{e_w \in S'} \{\Delta'(e_w)\} \tag{5}$$

Correspondingly, the move gain of $swap(p, q)$ can be finally computed as:

$$\Delta f(S, p, q) = f(S') - f(S) \tag{6}$$

With the help of this fast updating strategy, we can calculate $\Delta f(S, p, q)$ in $\mathcal{O}(m)$ because one only needs to check the $m - 1$ elements adjacent to the removed element $e_p$ in $S$ and the $m - 1$ elements adjacent to the added element $e_q$ in $S$. Since the total number of the neighboring solutions evaluated is $m(n - m)$, therefore the computational complexity to evaluate the whole neighborhood is $\mathcal{O}(m^2 n)$.

To explore the neighborhood, the *descent_based_neighborhood_search()* procedure uses the *best improvement strategy*. In other words, the best improving neighbor solution (with the smallest negative move gain) is selected at each iteration (ties are broken at random). After each solution transition, the

search is resumed from the new solution. When no improving neighbor solution exists in the neighborhood, the current solution is a local optimum. In this case, the *descent_based_neighborhood_search()* procedure terminates and returns the last solution as its output. Finally, after each swap operation, the $\Delta$ value of each element $e_i \in N$ is updated, which is achieved in $\mathcal{O}(n)$.

### *3.4 Local optima exploring phase*

Obviously, the descent-based neighborhood search phase will quickly fall into a local optimum because it only accepts improving solutions during the search. To intensify the search around the attained local optimum and discover other solutions of higher quality, we introduce the *local_optima_exploring()* procedure, which iterates through a moderate perturbation operation and the descent-based neighborhood search procedure (Algorithm 5).

**Algorithm 5** local_optima_exploring() procedure

**Input**: a starting solution $S$ and the given search depth $nbr_{max}$
**Output**: the best solution $S^*$ found during the current local optima exploring phase
$S^* \leftarrow S$ /* record the best solution found so far */
$nbr \leftarrow 0$
**while** $nbr < nbr_{max}$ **do**
  $S \leftarrow$ weak_perturb_operator($S$) /* perform a weak perturb operation */
  $S \leftarrow$ descent_based_neighborhood_search($S$) /* attain a local optimum */
  // update the best solution found
  **if** $f(S) < f(S^*)$ **then**
    $S^* \leftarrow S$
    $nbr \leftarrow 0$
  **else**
    $nbr \leftarrow nbr + 1$
  **end if**
**end while**

The *local_optima_exploring()* procedure uses the principle of deterministic tournament selection to perturb (slightly) the input local optimal solution $S$ with the *weak_perturb_operator()*. At each perturbation step, we first generate at random $n+1$ candidate neighbor solutions of the incumbent solution and then select the best one among these solutions to replace the incumbent solution. The *weak_perturb_operator()* repeats $p_w$ times ($p_w$ is a parameter called *weak perturbation strength*), and returns the last perturbed solution which serves as the starting point of the descent-based neighborhood search. It is clear that a small (large) $p_w$ leads to a perturbed solution which is close to (far away from) the input solution $S$. In this work, we set $p_w = 2, 3$.

Starting from the perturbed solution delivered by the *weak_perturb_operator()*, the *descent_based_neighborhood_search()* procedure is run to attain a new lo-

cal optimum, which becomes the incumbent solution of next iteration of the current local optima exploring phase. The best local optimum $S^*$ found during the local optima exploring phase is updated each time a new local optimum better than the recorded $S^*$ is encountered. The *local_optima_exploring()* procedure terminates when the recorded best local optimum $S^*$ cannot be updated for $nbr_{max}$ consecutive iterations ($nbr_{max}$ is a parameter called *search depth*), indicating that the region around the initial input solution $S$ is exhausted and the search needs to move into a more distant region, which is the purpose of the local optima escaping procedure described in the next section.

### *3.5 Local optima escaping phase*

To move away from the best local optimum $S^*$ found by *local_optima_exploring()*, we call for the *local_optima_escaping()* procedure which applies a strong perturbation mechanism. Specifically, the *local_optima_escaping()* procedure takes $S^*$ as its input, and then randomly performs $p_s$ swap operations. $p_s$, called strong perturbation strength, is defined by $p_s = \alpha \times n/m$, where $\alpha \in [1.0, 2.0)$ is a parameter called *strong perturbation coefficient*. Since the objective variations are not considered during the perturbation operations, the perturbed solution may be greatly different from the input local optimum. This is particularly true with large $p_s$ value (e.g., $p_s > 10$), which definitively helps the search to jump into a distant search region.

## 4 Computational experiments

This section presents a performance assessment of the ILS_MinDiff algorithm. For this purpose, we carry out an extensive experimental comparison between the proposed algorithm and two most recent and best performing algorithms (GRASP_EPR [13] and VNS_MinDiff [28]), based on six data sets of 190 benchmark instances.

### *4.1 Benchmark instances*

*MDPLIB* [1] provides a comprehensive set of instances which are widely used for testing algorithms for diversity and dispersion problems. By excluding the small and easy instances, the remaining 190 benchmark instances tested in this work include the following three types and are classified into six data sets.

[1] `http://www.optsicom.es/mindiff/`

- **SOM** (*SOM-b*): This data set includes 20 test instances whose sizes range from $(n, m) = (100, 10)$ to $(n, m) = (500, 200)$. The instances of this set were created with a generator developed by Silva et al. [31].
- **GKD** (*GKD-b* and *GKD-c*): These two data sets include 70 test instances whose sizes range from $(n, m) = (25, 2)$ to $(n, m) = (500, 50)$. The distance matrices are built by calculating the Euclidean distance between each pair of randomly generated points from the square $[0, 10] \times [0, 10]$. These instances were introduced by Glover et al. [18] and generated by Duarte and Martí [12], and Martí et al. [26].
- **MDG** (*MDG-a*, *MDG-b* and *MDG-c*): The whole data set is composed of 100 test instances whose sizes range from $(n, m) = (500, 50)$ to $(n, m) = (3000, 600)$. The distance matrices in these instances are generated by selecting real numbers between 0 and 10 from an uniform distribution. These instances have been widely used in, e.g., Duarte and Martí [12], Palubeckis [29], and Martí et al. [27].

### *4.2 Experimental settings*

The ILS_MinDiff algorithm [2] was implemented in C++ and compiled using g++ compiler with the '-O2' flag. All experiments were carried out on an Intel Xeon E5440 processor with 2.83 GHz and 2 GB RAM under Linux operating system. Without using any compilation flag, running the DIMACS machine benchmark procedure dfmax.c [3] on our machine requires 0.44, 2.63 and 9.85 seconds to solve the benchmark graphs r300.5, r400.5 and r500.5 respectively.

Table 2
Parameter settings of the proposed ILS_MinDiff algorithm

| Parameters | Section | Description | Value |
|---|---|---|---|
| $t_{max}$ | 3.1 | time limit | $n$ |
| $nbr_{max}$ | 3.4 | search depth | 5 |
| $p_w$ | 3.4 | weak perturbation strength | $\{2, 3\}$ |
| $\alpha$ | 3.5 | strong perturbation coefficient | 1.0 |

Given its stochastic nature, ILS_MinDiff was independently executed, like [28], forty times with different random seeds on each test instance. Each run stops if the running time reaches the cut-off time limit ($t_{max}$). Following the literature [28], we set the time limit $t_{max}$ to $n$, where $n$ is the number of elements in the considered test instance. To run the ILS_MinDiff algorithm, there are three parameters to be determined, including search depth $nbr_{max}$, weak perturbation strength $p_w$ in the local optima exploring phase and strong perturbation

[2] The best solution certificates and the code of our ILS_MinDiff algorithm will be made available at `http://www.info.univ-angers.fr/pub/hao/mindiffdp.html`
[3] dfmax: `ftp://dimacs.rutgers.edu/pub/dsj/clique`

coefficient $\alpha$ in the local optima escaping phase. These parameters were fixed according to the experimental analysis of Section 5: $nbr_{max} = 5$ for instances of all six data sets; $p_w = 3$ for instances with $n < 500$ or $n = 500, n/m < 10$, and $p_w = 2$ for the remaining instances; $\alpha = 1.0$ for all instances. A detailed description of the parameter settings is provided in Table 2. It would be possible that fine tuning these parameters or using relaxed stopping conditions would lead to better results. As we show below, with the adopted parameter settings, ILS_MinDiff already performs very well relative to the state-of-the-art results.

### *4.3 Comparison with the state-of-the-art algorithms*

As indicated in the introduction, several main approaches have been recently proposed in the literature to solve *Min-Diff DP*, including mixed integer programming (MIP) and greedy randomized adaptive search procedure (GRASP) in 2009 [30], GRASP with exterior path relinking (GRASP_EPR) in 2015 [13] and variable neighborhood search (VNS_MinDiff) in 2016 [28]. The computational results reported in [13] indicate that GRASP_EPR dominates MIP and GRASP of [30]. In [28], it was shown that the latest VNS_MinDiff algorithm performs the best by updating the best-known solutions for 170 out of 190 benchmark instances which were previously established by GRASP_EPR of [13]. Consequently, we adopted GRASP_EPR and VNS_MinDiff as our main reference algorithms to assess the performance of the proposed ILS_MinDiff algorithm. Detailed computational results of GRASP_EPR and VNS_MinDiff were extracted from `http://www.optsicom.es/mindiff/` and `http://www.mi.sanu.ac.rs/~nenad/mddp/` respectively. GRASP_EPR was implemented in Java 7 and the experiments were conducted on an Intel Core i7 2600 (3.4 GHz) with 4GB of RAM. Each test instance was solved only one time with the time limit $t_{max} = n$. VNS_MinDiff was coded in C++ and run on a computer with an Intel Core i7 2600 CPU (3.4 GHz) and 16 GB of RAM. Each instance was solved forty times with the time limit $t_{max} = n$ per run.

Given that the compared algorithms were executed on different platforms with different configurations, it seems difficult to strictly compare the runtimes. Therefore, we use *solution quality* as the main criterion for our comparative studies. Nevertheless, we also report the computing times consumed by the compared algorithms, which can still provide some useful indications about the computational efficiency of each algorithm. We note the computers used to run GRASP_EPR and VNS_MinDiff are roughly 1.2 ($3.4/2.83 \approx 1.2$) times faster than our computer according to their CPU frequencies. So the stopping condition $t_{max} = n$ used is slightly more favorable for the reference algorithms than for our ILS_MinDiff algorithm.

The comparative results of the proposed ILS_MinDiff algorithm and the ref-

erence algorithms (VNS_MinDiff and GRASP_EPR) are presented in Tables 3-8. In these tables, column 1 gives the name of each instance (Instance). Columns 2-3 show the results of GRASP_EPR, i.e., the best objective value ($f_{best}$) attained during a single run and the CPU time ($t_{best}$) to attain the best objective value. Columns 4-7 report the results of VNS_MinDiff, including the best objective value ($f_{best}$), the average objective value ($f_{avg}$), the worst objective value ($f_{worst}$) and the average time ($t_{avg}$) to attain the best objective value of the forty runs (i.e., $t_{avg} = \sum_{i=1}^{40} t_{best}^i/40$ where $t_{best}^i$ is the time of $i$th run to attain its best objective value. The same statistics of the proposed ILS_MinDiff algorithm are shown in columns 8-11. For ILS_MinDiff, we also report the standard deviation ($\sigma$), while this information is not available for VNS_MinDiff and not applicable for GRASP_EPR (since it was run once). The last column indicates the difference $\Delta f_{best}$ between the best solution values found by ILS_MinDiff and the best known solution values reported in the literature (a negative/positive value indicates an improved/worsen result). The best values among the results of the compared algorithms are highlighted in bold. The last row of each table indicates the average value of each comparison indicator. We complete this information by including between the parentheses the number of instances for which an algorithm obtained the best performance compared to the other algorithms (if two algorithms report the same best performance on one instance, the instance was counted 0.5 for each algorithm).

To analyze the experimental results, we resort to the well-known *two-tailed sign test* [10] to check the significant difference on each comparison indicator between the compared algorithms. The two-tailed sign test is a popular technique to compare the overall performances of algorithms by counting the number of instances for which an algorithm is the overall winner. When two algorithms are compared, the corresponding null-hypothesis is that two algorithms are equivalent. The null-hypothesis is accepted if and only if each algorithm wins on approximately $X/2$ out of $X$ instances. Since tied matches support the null-hypothesis [10], we split them evenly between the two compared algorithms, i.e., each one wins 0.5.

We first compare GRASP_EPR with VNS_MinDiff and ILS_MinDiff. Since GRASP_EPR was run only *once* to solve each instance while VNS_MinDiff and ILS_MinDiff were run *forty times* per instance, it is unfair to compare the best results of GRASP_EPR with the best results of VNS_MinDiff and ILS_MinDiff. Instead, we can compare the *best* objective values of GRASP_EPR with the *average* objective values of VNS_MinDiff and ILS_MinDiff since the average result can be reasonably considered as a representative single run. As we observe from Tables 3-8, both VNS_MinDiff and ILS_MinDiff dominate GRASP_EPR for all six data sets according to this indicator. The results also indicate that even the *worst* results of VNS_MinDiff and ILS_MinDiff (in particular, those of ILS_MinDiff) are better than the best results of GRASP_EPR apart from a few exceptions. Concerning the computing efficiency, given that the com-

pared algorithms obtain solutions of quite different quality, it is not meaningful to compare their computing times. Based on this comparison, we decided to exclude GRASP_EPR for further comparisons and focus our study on a performance assessment of ILS_MinDiff with respect to VNS_MinDiff.

Based on the results of ILS_MinDiff and VNS_MinDiff reported in Tables 3-8, we observe first that the two-tailed sign test rejected the null hypothesis in all six data sets, suggesting these two algorithms do not have an equal performance. At a significance level of 0.05, the *Critical Values* ($CV$) of the two-tailed sign test were respectively $CV_{0.05}^{20} = 15$, $CV_{0.05}^{40} = 27$, and $CV_{0.05}^{50} = 32$ when the number of instances in each data set is $X = 20$, $X = 40$, and $X = 50$. This means that algorithm A is significantly better than algorithm B if A wins at least $CV_{0.05}^{X}$ instances for a data set of $X$ instances. Below, we analyze the performances of ILS_MinDiff and VNS_MinDiff for each of the six data sets.

From Table 3 which concerns the 20 instances of data set *SOM-b*, it can be observed that the proposed ILS_MinDiff algorithm competes very favorably with the reference VNS_MinDiff algorithm in all listed indicators. For this data set, ILS_MinDiff easily dominates VNS_MinDiff by improving the best-known results for 16 out of 20 instances and matching the best-known upper bounds for the remaining 4 instances. In addition, ILS_MinDiff outperforms the reference algorithm in terms of the average solution value as well as the worst solution value. To achieve these best results, ILS_MinDiff also spends much less time than that of the VNS_MinDiff algorithm. A small standard deviation over forty runs demonstrates that ILS_MinDiff has a stable performance. In this case, $CV_{0.05}^{20} = 15$, the two-tailed sign test confirms the statistical significance of the differences between these two algorithms in all comparison indicators on solution values.

Table 4 displays the comparative results of GRASP_EPR, VNS_MinDiff and ILS_MinDiff on the 50 instances of data set *GKD-b*. As we can see from the table, the proposed ILS_MinDiff algorithm achieves new best solutions (improved upper bounds) for 13 out of 50 instances and matches the best-known solutions for the remaining 37 instances. Moreover, compared to VNS_MinDiff, ILS_MinDiff respectively obtains a better average solution value and worst solution value for 36 and 32 out of 50 instances. It is worth mentioning that ILS_MinDiff usually achieves a small standard deviation $\sigma$, and even $\sigma = 0.0$ for 27 out of 50 instances, which shows that ILS_MinDiff performs stably on this data set. Furthermore, the two-tailed sign test confirms that ILS_MinDiff is significantly better than VNS_MinDiff in terms of the average and worst values, while for the best solution value, the number of instances wined by ILS_MinDiff is only slightly smaller than the critical value, i.e., $31.5 < CV_{0.05}^{50} = 32$.

Table 3. Comparative results of the proposed ILS_MinDiff algorithm and reference algorithms on *SOM-b*.

| | GRASP_EPR [13](2015) | | VNS_MinDiff [28](2016) | | | | ILS_MinDiff | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | $f_{best}$ | $t_{best}$ | $f_{best}$ | $f_{avg}$ | $f_{worst}$ | $t_{avg}$ | $f_{best}$ | $f_{avg}$ | $f_{worst}$ | $t_{avg}$ | $\sigma$ | $\Delta f_{best}$ |
| SOM-b_1_n100_m10 | 2.00 | 0.70 | 1.00 | 1.00 | **1.00** | 1.24 | **0.00** | **0.83** | **1.00** | 12.06 | 0.38 | -1.00 |
| SOM-b_2_n100_m20 | 6.00 | 3.04 | **4.00** | 4.50 | **5.00** | 23.80 | **4.00** | **4.40** | **5.00** | 29.62 | 0.49 | 0.00 |
| SOM-b_3_n100_m30 | 10.00 | 5.80 | 8.00 | 8.60 | **9.00** | 18.06 | **7.00** | **7.88** | **9.00** | 23.82 | 0.40 | -1.00 |
| SOM-b_4_n100_m40 | 13.00 | 8.72 | 12.00 | 12.20 | 13.00 | 30.55 | **10.00** | **10.98** | **12.00** | 34.41 | 0.57 | -2.00 |
| SOM-b_5_n200_m20 | 5.00 | 5.93 | **3.00** | **3.90** | **4.00** | 68.52 | **3.00** | 4.00 | 5.00 | 64.52 | 0.39 | 0.00 |
| SOM-b_6_n200_m40 | 13.00 | 24.92 | 10.00 | 10.50 | **11.00** | 87.63 | **9.00** | **10.40** | **11.00** | 47.47 | 0.54 | -1.00 |
| SOM-b_7_n200_m60 | 19.00 | 51.93 | 16.00 | 16.70 | 18.00 | 75.09 | **15.00** | **15.88** | **17.00** | 67.74 | 0.51 | -1.00 |
| SOM-b_8_n200_m80 | 27.00 | 74.15 | 22.00 | 24.00 | 26.00 | 58.43 | **19.00** | **21.50** | **23.00** | 76.80 | 0.78 | -3.00 |
| SOM-b_9_n300_m30 | 9.00 | 23.38 | 7.00 | 7.40 | **8.00** | 83.01 | **6.00** | **7.03** | **8.00** | 84.45 | 0.42 | -1.00 |
| SOM-b_10_n300_m60 | 17.00 | 88.86 | 15.00 | 16.20 | 17.00 | 97.65 | **14.00** | **15.55** | **16.00** | 103.76 | 0.59 | -1.00 |
| SOM-b_11_n300_m90 | 27.00 | 173.61 | 22.00 | 24.10 | 26.00 | 94.03 | **21.00** | **22.98** | **24.00** | 107.44 | 0.72 | -1.00 |
| SOM-b_12_n300_m120 | 36.00 | 300.00 | 29.00 | 31.90 | 34.00 | 135.42 | **28.00** | **30.13** | **32.00** | 110.24 | 0.95 | -1.00 |
| SOM-b_13_n400_m40 | 12.00 | 53.34 | 10.00 | 10.40 | 11.00 | 80.32 | **9.00** | **9.85** | **10.00** | 118.24 | 0.36 | -1.00 |
| SOM-b_14_n400_m80 | 24.00 | 239.43 | **19.00** | 21.30 | 23.00 | 165.73 | **19.00** | **20.53** | **21.00** | 123.43 | 0.55 | 0.00 |
| SOM-b_15_n400_m120 | 38.00 | 400.00 | 30.00 | 31.70 | 34.00 | 204.59 | **28.00** | **30.03** | **32.00** | 160.33 | 0.76 | -2.00 |
| SOM-b_16_n400_m160 | 54.00 | 400.00 | 40.00 | 43.40 | 47.00 | 255.04 | **36.00** | **38.73** | **41.00** | 188.42 | 1.27 | -4.00 |
| SOM-b_17_n500_m50 | 13.00 | 114.40 | 12.00 | 12.80 | **13.00** | 195.87 | **11.00** | **12.33** | **13.00** | 158.23 | 0.52 | -1.00 |
| SOM-b_18_n500_m100 | 26.00 | 500.00 | **23.00** | 25.10 | 27.00 | 232.08 | **23.00** | **24.70** | **26.00** | 238.64 | 0.87 | 0.00 |
| SOM-b_19_n500_m150 | 47.00 | 500.00 | 36.00 | 39.60 | 45.00 | 248.05 | **34.00** | **37.00** | **38.00** | 224.23 | 1.00 | -2.00 |
| SOM-b_20_n500_m200 | 69.00 | 500.00 | 49.00 | 56.40 | 63.00 | 268.57 | **43.00** | **47.10** | **50.00** | 264.56 | 1.50 | -6.00 |
| Average value (wins) | 23.35 | 173.41 | 18.40(2) | 20.09(1) | 21.75(4) | 121.47 | **16.95**(18) | **18.59**(19) | **19.70**(16) | 111.92 | 0.68 | -1.45 |

Table 4. Comparative results of the proposed ILS_MinDiff algorithm and reference algorithms on *GKD-b*.

| | GRASP_EPR [13](2015) | | VNS_MinDiff [28](2016) | | | | ILS_MinDiff | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | $f_{best}$ | $t_{best}$ | $f_{best}$ | $f_{avg}$ | $f_{worst}$ | $t_{avg}$ | $f_{best}$ | $f_{avg}$ | $f_{worst}$ | $t_{avg}$ | $\sigma$ | $\Delta f_{best}$ |
| GKD-b_1_n25_m2 | **0.00** | 0.00 | **0.00** | **0.00** | **0.00** | 0.00 | **0.00** | **0.00** | **0.00** | 0.00 | 0.00 | 0.00 |
| GKD-b_2_n25_m2 | **0.00** | 0.00 | **0.00** | **0.00** | **0.00** | 0.00 | **0.00** | **0.00** | **0.00** | 0.00 | 0.00 | 0.00 |
| GKD-b_3_n25_m2 | **0.00** | 0.02 | **0.00** | **0.00** | **0.00** | 0.00 | **0.00** | **0.00** | **0.00** | 0.00 | 0.00 | 0.00 |
| GKD-b_4_n25_m2 | **0.00** | 0.02 | **0.00** | **0.00** | **0.00** | 0.00 | **0.00** | **0.00** | **0.00** | 0.00 | 0.00 | 0.00 |
| GKD-b_5_n25_m2 | **0.00** | 0.02 | **0.00** | **0.00** | **0.00** | 0.00 | **0.00** | **0.00** | **0.00** | 0.00 | 0.00 | 0.00 |
| GKD-b_6_n25_m7 | **12.72** | 0.17 | **12.72** | **12.72** | **12.72** | 0.00 | **12.72** | **12.72** | **12.72** | 0.12 | 0.00 | 0.00 |
| GKD-b_7_n25_m7 | **14.10** | 0.16 | **14.10** | **14.10** | **14.10** | 0.00 | **14.10** | **14.10** | **14.10** | 5.26 | 0.00 | 0.00 |
| GKD-b_8_n25_m7 | **16.76** | 0.16 | **16.76** | **16.76** | **16.76** | 0.00 | **16.76** | **16.76** | **16.76** | 6.37 | 0.00 | 0.00 |
| GKD-b_9_n25_m7 | **17.07** | 0.17 | **17.07** | **17.07** | **17.07** | 0.00 | **17.07** | **17.07** | **17.07** | 1.07 | 0.00 | 0.00 |
| GKD-b_10_n25_m7 | **23.27** | 0.31 | **23.27** | **23.27** | **23.27** | 0.00 | **23.27** | **23.27** | **23.27** | 5.72 | 0.00 | 0.00 |
| GKD-b_11_n50_m5 | **1.93** | 0.19 | **1.93** | **1.93** | **1.93** | 0.01 | **1.93** | **1.93** | **1.93** | 12.52 | 0.00 | 0.00 |
| GKD-b_12_n50_m5 | 2.12 | 0.17 | **2.05** | **2.05** | **2.05** | 0.04 | **2.05** | **2.05** | **2.05** | 3.90 | 0.00 | 0.00 |
| GKD-b_13_n50_m5 | **2.36** | 0.19 | **2.36** | **2.36** | **2.36** | 0.02 | **2.36** | **2.36** | **2.36** | 1.79 | 0.00 | 0.00 |
| GKD-b_14_n50_m5 | **1.66** | 0.19 | **1.66** | **1.66** | **1.66** | 0.01 | **1.66** | **1.66** | **1.66** | 10.80 | 0.00 | 0.00 |
| GKD-b_15_n50_m5 | **2.85** | 0.19 | **2.85** | **2.85** | **2.85** | 0.00 | **2.85** | **2.85** | **2.85** | 5.25 | 0.00 | 0.00 |
| GKD-b_16_n50_m15 | **42.75** | 1.39 | **42.75** | **42.75** | **42.75** | 0.01 | **42.75** | **42.75** | **42.75** | 13.74 | 0.00 | 0.00 |
| GKD-b_17_n50_m15 | **48.11** | 1.61 | **48.11** | **48.11** | **48.11** | 0.00 | **48.11** | **48.11** | **48.11** | 8.83 | 0.00 | 0.00 |
| GKD-b_18_n50_m15 | **43.20** | 1.34 | **43.20** | **43.20** | **43.20** | 0.02 | **43.20** | **43.20** | **43.20** | 9.53 | 0.00 | 0.00 |
| GKD-b_19_n50_m15 | **46.41** | 1.36 | **46.41** | **46.41** | **46.41** | 0.25 | **46.41** | **46.41** | **46.41** | 13.40 | 0.00 | 0.00 |
| GKD-b_20_n50_m15 | **47.72** | 1.27 | **47.72** | **47.72** | **47.72** | 0.00 | **47.72** | **47.72** | **47.72** | 16.15 | 0.00 | 0.00 |
| GKD-b_21_n100_m10 | 13.83 | 1.17 | **9.33** | 9.38 | 9.50 | 32.65 | **9.33** | **9.34** | **9.43** | 30.15 | 0.02 | 0.00 |
| GKD-b_22_n100_m10 | 13.66 | 1.17 | **8.04** | 9.01 | 10.22 | 38.35 | **8.04** | **8.46** | **9.58** | 45.43 | 0.41 | 0.00 |
| GKD-b_23_n100_m10 | 15.35 | 1.08 | **6.91** | 7.63 | 9.10 | 29.66 | **6.91** | **7.16** | **8.64** | 52.55 | 0.42 | 0.00 |
| GKD-b_24_n100_m10 | 8.64 | 1.20 | 7.17 | 7.65 | **7.94** | 43.32 | **5.79** | **7.24** | 8.94 | 53.52 | 0.76 | -1.38 |
| GKD-b_25_n100_m10 | 17.20 | 1.33 | **6.91** | 8.65 | 10.43 | 46.92 | 6.92 | **8.43** | **10.28** | 58.28 | 0.91 | +0.01 |
| GKD-b_26_n100_m30 | 168.73 | 9.44 | **159.19** | 160.37 | 166.31 | 9.41 | **159.19** | **159.19** | **159.19** | 43.59 | 0.00 | 0.00 |
| GKD-b_27_n100_m30 | 127.10 | 9.72 | **124.17** | 125.05 | 127.10 | 25.58 | **124.17** | **124.17** | **124.17** | 35.40 | 0.00 | 0.00 |
| GKD-b_28_n100_m30 | **106.38** | 10.42 | **106.38** | **106.38** | **106.38** | 16.34 | **106.38** | **106.38** | **106.38** | 31.37 | 0.00 | 0.00 |
| GKD-b_29_n100_m30 | 137.45 | 10.05 | **135.85** | **135.85** | **135.85** | 33.27 | **135.85** | **135.85** | **135.85** | 39.90 | 0.00 | 0.00 |
| GKD-b_30_n100_m30 | 127.48 | 9.28 | **127.27** | 127.27 | 127.27 | 86.29 | **127.27** | **127.27** | **127.27** | 58.78 | 0.00 | 0.00 |
| GKD-b_39_n125_m37 | 168.59 | 18.43 | 159.48 | 168.84 | 181.13 | 62.17 | **155.39** | **158.74** | **160.83** | 60.80 | 1.60 | -4.09 |
| GKD-b_40_n125_m37 | 178.19 | 18.18 | 174.34 | 186.65 | 205.13 | 56.29 | **172.80** | **178.58** | **181.78** | 60.16 | 1.63 | -1.54 |

Continued on next page

Table 4 – continued from previous page

| | GRASP_EPR [13](2015) | | VNS_MinDiff [28](2016) | | | | ILS_MinDiff | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | $f_{best}$ | $t_{best}$ | $f_{best}$ | $f_{avg}$ | $f_{worst}$ | $t_{avg}$ | $f_{best}$ | $f_{avg}$ | $f_{worst}$ | $t_{avg}$ | $\sigma$ | $\Delta f_{best}$ |
| GKD-b_41_n150_m15 | 23.35 | 4.39 | 17.40 | 19.60 | **20.98** | 64.52 | **15.85** | **19.56** | 22.44 | 69.33 | 1.54 | -1.55 |
| GKD-b_42_n150_m15 | 26.79 | 4.59 | 18.20 | 19.42 | **20.94** | 73.41 | **13.96** | **17.96** | 21.18 | 76.20 | 1.63 | -4.24 |
| GKD-b_43_n150_m15 | 26.75 | 4.15 | 15.57 | 17.66 | 19.93 | 58.30 | **11.83** | **16.56** | **19.56** | 62.96 | 1.76 | -3.74 |
| GKD-b_44_n150_m15 | 25.94 | 4.32 | 15.16 | **17.58** | **19.28** | 87.71 | **11.74** | 17.73 | 21.33 | 71.02 | 1.87 | -3.42 |
| GKD-b_45_n150_m15 | 27.77 | 4.36 | 15.23 | 18.38 | 21.77 | 65.09 | **12.84** | **17.83** | **20.50** | 75.93 | 1.87 | -2.39 |
| GKD-b_46_n150_m45 | 227.75 | 34.37 | **207.81** | 225.89 | 258.53 | 41.74 | **207.81** | **209.38** | **214.56** | 72.08 | 2.79 | 0.00 |
| GKD-b_47_n150_m45 | 228.60 | 34.57 | 214.42 | 228.62 | 259.86 | 56.65 | **211.77** | **214.14** | **218.14** | 74.34 | 1.61 | -2.65 |
| GKD-b_48_n150_m45 | 226.75 | 30.27 | 180.00 | 221.04 | 239.52 | 48.94 | **177.29** | **184.01** | **192.36** | 69.80 | 3.71 | -2.71 |
| GKD-b_49_n150_m45 | 226.41 | 36.04 | 205.39 | 227.51 | 288.07 | 37.33 | **197.88** | **201.12** | **206.75** | 80.44 | 3.07 | -7.51 |
| GKD-b_50_n150_m45 | 248.86 | 33.04 | **220.76** | 245.17 | 279.55 | 80.03 | **220.76** | **225.78** | **234.81** | 78.84 | 4.54 | 0.00 |
| Average value (wins) | 66.47 | 7.18 | 60.26(19) | 64.36(14) | 70.76(18) | 26.28 | **59.51**(31) | **60.82**(36) | **62.27**(32) | 36.60 | 0.72 | -0.75 |

Table 5. Comparative results of the proposed ILS_MinDiff algorithm and reference algorithms on *GKD-c*.

| | GRASP_EPR [13](2015) | | VNS_MinDiff [28](2016) | | | | ILS_MinDiff | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | $f_{best}$ | $t_{best}$ | $f_{best}$ | $f_{avg}$ | $f_{worst}$ | $t_{avg}$ | $f_{best}$ | $f_{avg}$ | $f_{worst}$ | $t_{avg}$ | $\sigma$ | $\Delta f_{best}$ |
| GKD-c_1_n500_m50 | 16.85 | 186.20 | 10.09 | 12.09 | 13.63 | 344.96 | **9.85** | **10.85** | **11.93** | 221.70 | 0.54 | -0.24 |
| GKD-c_2_n500_m50 | 16.53 | 189.93 | 10.79 | 12.98 | 15.41 | 351.47 | **10.06** | **11.64** | **12.79** | 202.54 | 0.71 | -0.73 |
| GKD-c_3_n500_m50 | 18.50 | 181.71 | **8.84** | 11.81 | 16.29 | 366.57 | 9.87 | **11.71** | **13.26** | 262.49 | 0.68 | +1.03 |
| GKD-c_4_n500_m50 | 18.87 | 173.69 | **9.42** | 12.82 | 15.99 | 436.28 | 9.55 | **11.51** | **13.60** | 262.46 | 0.70 | +0.13 |
| GKD-c_5_n500_m50 | 18.45 | 182.91 | 11.26 | 13.53 | 16.00 | 235.73 | **10.36** | **12.03** | **13.11** | 267.60 | 0.68 | -0.90 |
| GKD-c_6_n500_m50 | 17.92 | 183.83 | 10.63 | 12.14 | 14.49 | 367.35 | **10.21** | **11.36** | **12.37** | 254.12 | 0.56 | -0.42 |
| GKD-c_7_n500_m50 | 17.54 | 173.60 | 11.58 | 13.71 | 16.65 | 158.51 | **10.47** | **12.27** | **13.45** | 253.01 | 0.66 | -1.11 |
| GKD-c_8_n500_m50 | 19.86 | 186.61 | 11.31 | 13.79 | 20.50 | 341.60 | **9.89** | **11.55** | **12.88** | 287.74 | 0.70 | -1.42 |
| GKD-c_9_n500_m50 | 17.96 | 169.37 | 10.45 | 13.71 | 17.64 | 320.03 | **9.69** | **11.29** | **12.61** | 262.99 | 0.68 | -0.76 |
| GKD-c_10_n500_m50 | 17.10 | 180.95 | **9.21** | 13.94 | 18.45 | 358.62 | 10.99 | **12.14** | **13.24** | 288.20 | 0.56 | +1.78 |
| GKD-c_11_n500_m50 | 15.77 | 184.50 | 11.03 | 12.39 | 14.59 | 370.01 | **9.03** | **10.99** | **12.10** | 206.79 | 0.76 | -2.00 |
| GKD-c_12_n500_m50 | 17.71 | 179.79 | **9.48** | 12.98 | 15.06 | 303.82 | 9.82 | **11.42** | **12.60** | 203.12 | 0.63 | +0.34 |
| GKD-c_13_n500_m50 | 17.04 | 184.04 | 10.04 | 12.51 | 15.38 | 351.28 | **9.94** | **12.10** | **13.86** | 261.22 | 0.78 | -0.10 |
| GKD-c_14_n500_m50 | 19.27 | 181.15 | 11.28 | 12.88 | 15.60 | 274.72 | **9.24** | **11.54** | **12.48** | 300.43 | 0.62 | -2.04 |
| GKD-c_15_n500_m50 | 17.65 | 177.48 | 10.85 | 13.78 | 17.59 | 278.53 | **9.53** | **12.11** | **13.51** | 219.81 | 0.88 | -1.32 |
| GKD-c_16_n500_m50 | 16.32 | 179.78 | **8.39** | 12.30 | 15.11 | 349.12 | 10.04 | **11.62** | **12.51** | 268.00 | 0.58 | +1.65 |
| GKD-c_17_n500_m50 | 17.56 | 180.31 | 10.14 | 11.52 | 13.29 | 350.12 | **9.90** | **11.16** | **12.31** | 261.73 | 0.60 | -0.24 |
| GKD-c_18_n500_m50 | 19.03 | 180.01 | **9.77** | 13.47 | 17.06 | 289.34 | 10.56 | **12.04** | **13.03** | 217.42 | 0.56 | +0.79 |
| GKD-c_19_n500_m50 | 18.15 | 192.12 | 11.11 | 13.56 | 16.36 | 276.61 | **10.25** | **11.73** | **12.67** | 219.49 | 0.62 | -0.86 |
| GKD-c_20_n500_m50 | 18.53 | 182.48 | 10.44 | 12.82 | 15.12 | 354.73 | **10.10** | **11.52** | **12.74** | 273.15 | 0.62 | -0.34 |
| Average value (wins) | 17.83 | 181.52 | 10.31(6) | 12.94(0) | 16.01(0) | 323.97 | **9.97**(14) | **11.63**(20) | **12.85**(20) | 249.70 | 0.65 | -0.34 |

Table 5 presents the comparative results of GRASP_EPR, VNS_MinDiff and ILS_MinDiff on the 20 instances of data set *GKD-c*. From Table 5, we observe that ILS_MinDiff achieves a new improved solution for 14 out of 20 instances. ILS_MinDiff fully dominates the reference algorithm by obtaining a better average solution value and worst solution value for all 20 instances. In addition, ILS_MinDiff needs much less CPU times to achieve these results for almost all 20 instances. Although no significant difference is observed on the best solution values between ILS_MinDiff and VNS_MinDiff ($14 < CV_{0.05}^{20} = 15$), ILS_MinDiff significantly performs better than VNS_MinDiff in terms of the average solution value and the worst solution value.

Table 6 shows the comparative results of ILS_MinDiff with the reference algorithms on the 40 instances of data set *MDG-a*. These results show that ILS_MinDiff again outperforms the reference algorithms. Specifically, in terms of the best solution value, ILS_MinDiff finds a new best solution for 32 out of 40 instances and matches the best-known solutions for 2 instances. Concerning the average solution value, ILS_MinDiff remains competitive, achieving a better average solution value for 30 out of 40 instances. In addition, ILS_MinDiff also wins much more instances than VNS_MinDiff, i.e., 21 instances for the worst solution value. With the two-tailed sign test, at a significance level of 0.05, we find that ILS_MinDiff is significantly better than VNS_MinDiff algorithm in terms of the best solution value ($33 > CV_{0.05}^{40} = 27$) and average solution value ($30 > CV_{0.05}^{40} = 27$).

Table 7 on the 40 instances of data set *MDG-b* demonstrates that ILS_MinDiff achieves a highly competitive performance by obtaining a new best solution for 37 out of 40 instances. Moreover, ILS_MinDiff achieves a better average solution value and worst solution value, and respectively wins 38 and 30 out of 40 instances with respect to VNS_MinDiff. The two-tailed sign test confirms that the differences of these two algorithms are statistically significant. ILS_MinDiff uses less time than VNS_MinDiff for 26 out of 40 instances. It is only sightly smaller than the critical value of the statistical test at the significance level of 0.05, i.e., $26 < CV_{0.05}^{40} = 27$. It is worth noting that the ILS_MinDiff algorithm achieves these results with a relatively large average standard deviation 98.97, suggesting that ILS_MinDiff is less stable on this data set than on the previous data sets.

Table 8 summaries the comparative results of ILS_MinDiff, GRASP_EPR and VNS_MinDiff on the 20 instances of data set *MDG-c* [4] . From this table, we observe that our ILS_MinDiff algorithm overall performs competitively. In terms of the best solution value, ILS_MinDiff significantly outperforms the reference

[4] We observe that VNS_MinDiff (and GRASP_EPR) report abnormal computing times for several instances, exceeding the allowed time limit ($t_{max} = n$). No information is available to explain these anomalies.

Table 6. Comparative results of the proposed ILS_MinDiff algorithm and reference algorithms on *MDG-a*.

| | GRASP_EPR [13](2015) | | VNS_MinDiff [28](2016) | | | | ILS_MinDiff | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | $f_{best}$ | $t_{best}$ | $f_{best}$ | $f_{avg}$ | $f_{worst}$ | $t_{avg}$ | $f_{best}$ | $f_{avg}$ | $f_{worst}$ | $t_{avg}$ | $\sigma$ | $\Delta f_{best}$ |
| MDG-a_1_n500_m50 | 13.53 | 179.47 | 11.34 | 12.26 | **12.67** | 151.84 | **11.11** | **12.09** | 12.99 | 259.36 | 0.43 | -0.23 |
| MDG-a_2_n500_m50 | 12.99 | 176.56 | 11.67 | 12.45 | 12.94 | 189.10 | **11.00** | **12.10** | **12.86** | 227.06 | 0.40 | -0.67 |
| MDG-a_3_n500_m50 | 13.34 | 172.91 | 11.71 | 12.22 | 12.82 | 311.65 | **11.31** | **12.08** | **12.68** | 253.43 | 0.32 | -0.40 |
| MDG-a_4_n500_m50 | 13.41 | 178.02 | 11.56 | 12.34 | 12.94 | 258.35 | **11.35** | **12.05** | **12.65** | 253.37 | 0.34 | -0.21 |
| MDG-a_5_n500_m50 | 13.50 | 164.69 | 12.05 | 12.50 | 12.77 | 233.59 | **10.75** | **12.08** | **12.67** | 275.97 | 0.41 | -1.30 |
| MDG-a_6_n500_m50 | 12.95 | 180.56 | **10.87** | 12.15 | 12.74 | 328.59 | 11.11 | **12.07** | **12.72** | 255.47 | 0.40 | +0.24 |
| MDG-a_7_n500_m50 | 13.09 | 173.27 | 10.95 | 12.14 | 13.17 | 292.69 | **10.64** | **12.08** | **12.7**6 | 234.95 | 0.46 | -0.31 |
| MDG-a_8_n500_m50 | 13.89 | 170.31 | 11.80 | 12.41 | **13.00** | 204.47 | **10.16** | **11.99** | **13.00** | 251.16 | 0.52 | -1.64 |
| MDG-a_9_n500_m50 | 13.61 | 176.66 | 11.54 | 12.37 | 12.80 | 207.14 | **10.98** | **11.98** | **12.61** | 250.43 | 0.40 | -0.56 |
| MDG-a_10_n500_m50 | 12.56 | 175.50 | 11.60 | 12.33 | 13.00 | 248.57 | **11.04** | **12.01** | **12.57** | 252.52 | 0.36 | -0.56 |
| MDG-a_11_n500_m50 | 13.21 | 174.29 | 11.25 | 12.12 | **12.68** | 117.75 | **10.83** | **11.96** | 12.71 | 237.47 | 0.47 | -0.42 |
| MDG-a_12_n500_m50 | 13.01 | 182.68 | 12.17 | 12.53 | 12.87 | 251.91 | **11.12** | **12.03** | **12.69** | 244.03 | 0.37 | -1.05 |
| MDG-a_13_n500_m50 | 12.70 | 170.06 | 12.05 | 12.41 | 12.99 | 298.51 | **10.63** | **12.00** | **12.74** | 216.85 | 0.46 | -1.42 |
| MDG-a_14_n500_m50 | 12.89 | 181.77 | 11.60 | 12.42 | 13.06 | 164.87 | **10.77** | **11.94** | **12.79** | 289.92 | 0.41 | -0.83 |
| MDG-a_15_n500_m50 | 13.51 | 178.36 | 11.55 | 12.39 | 12.91 | 221.15 | **10.97** | **12.01** | **12.77** | 274.32 | 0.42 | -0.58 |
| MDG-a_16_n500_m50 | 13.19 | 176.83 | 12.15 | 12.64 | 13.12 | 240.76 | **10.65** | **12.04** | **12.76** | 242.01 | 0.43 | -1.50 |
| MDG-a_17_n500_m50 | 12.48 | 180.14 | 11.76 | 12.32 | **12.73** | 276.39 | **10.88** | **12.12** | 12.94 | 293.07 | 0.45 | -0.88 |
| MDG-a_18_n500_m50 | 11.49 | 169.06 | 11.95 | 12.42 | 12.90 | 317.89 | **10.88** | **12.12** | **12.60** | 286.69 | 0.35 | -1.07 |
| MDG-a_19_n500_m50 | 13.50 | 177.66 | **11.50** | **12.34** | 12.93 | 241.46 | 11.57 | **12.21** | 12.71 | 249.01 | 0.28 | +0.07 |
| MDG-a_20_n500_m50 | 13.20 | 175.63 | 11.66 | 12.18 | **12.60** | 253.48 | **11.10** | **12.17** | 13.20 | 284.12 | 0.42 | -0.56 |
| MDG-a_21_n2000_m200 | 68.00 | 2000.00 | **50.00** | **53.10** | **57.00** | 1359.44 | **50.00** | 53.43 | **57.00** | 1342.38 | 1.79 | 0.00 |
| MDG-a_22_n2000_m200 | 70.00 | 2000.01 | 51.00 | 53.60 | **56.00** | 1490.53 | **50.00** | **53.55** | 58.00 | 1446.83 | 2.20 | -1.00 |
| MDG-a_23_n2000_m200 | 63.00 | 2000.00 | 52.00 | 54.30 | **57.00** | 959.98 | **49.00** | **53.60** | 58.00 | 1380.25 | 2.49 | -3.00 |
| MDG-a_24_n2000_m200 | 63.00 | 2000.00 | **48.00** | **53.00** | **58.00** | 1348.31 | 50.00 | 53.63 | **58.00** | 1447.23 | 2.03 | +2.00 |
| MDG-a_25_n2000_m200 | 57.00 | 2000.00 | 51.00 | 54.30 | **58.00** | 1255.08 | **50.00** | **53.60** | **58.00** | 1420.26 | 1.95 | -1.00 |
| MDG-a_26_n2000_m200 | 68.00 | 2000.00 | **49.00** | **53.00** | **57.00** | 1136.47 | 50.00 | 53.58 | **57.00** | 1410.15 | 1.69 | +1.00 |
| MDG-a_27_n2000_m200 | 62.00 | 2000.00 | 50.00 | 54.70 | **58.00** | 1196.13 | **49.00** | **53.73** | **58.00** | 1406.19 | 2.18 | -1.00 |
| MDG-a_28_n2000_m200 | 64.00 | 2000.00 | **48.00** | 53.10 | **57.00** | 1280.44 | 50.00 | **52.98** | 59.00 | 1464.31 | 2.03 | +2.00 |
| MDG-a_29_n2000_m200 | 63.00 | 2000.01 | 51.00 | **53.00** | **56.00** | 1097.72 | **47.00** | 53.48 | 58.00 | 1542.65 | 2.61 | -4.00 |
| MDG-a_30_n2000_m200 | 65.00 | 2000.00 | 50.00 | **54.00** | **57.00** | 804.95 | **49.00** | 54.28 | 59.00 | 1469.70 | 2.93 | -1.00 |
| MDG-a_31_n2000_m200 | 67.00 | 2000.00 | 50.00 | 54.50 | 60.00 | 1084.71 | **49.00** | **53.88** | **58.00** | 1517.10 | 2.32 | -1.00 |
| MDG-a_32_n2000_m200 | 57.00 | 2000.00 | 51.00 | 54.60 | 61.00 | 1049.13 | **48.00** | **53.25** | **57.00** | 1383.72 | 1.84 | -3.00 |
| MDG-a_33_n2000_m200 | 67.00 | 2000.01 | 49.00 | **53.70** | 60.00 | 1315.45 | **48.00** | 53.80 | **59.00** | 1634.04 | 2.66 | -1.00 |
| MDG-a_34_n2000_m200 | 59.00 | 2000.00 | **49.00** | **53.30** | **57.00** | 1058.42 | **49.00** | 53.48 | 59.00 | 1388.50 | 2.23 | 0.00 |
| MDG-a_35_n2000_m200 | 67.00 | 2000.00 | 53.00 | 54.70 | **56.00** | 1020.57 | **48.00** | **54.08** | 59.00 | 1431.79 | 2.18 | -5.00 |
| MDG-a_36_n2000_m200 | 57.00 | 2000.00 | 51.00 | 54.10 | **57.00** | 1300.08 | **48.00** | **53.73** | 59.00 | 1439.09 | 2.41 | -3.00 |
| MDG-a_37_n2000_m200 | 57.00 | 2000.00 | 49.00 | **52.90** | **56.00** | 1294.62 | **48.00** | 53.85 | 58.00 | 1398.92 | 2.37 | -1.00 |
| MDG-a_38_n2000_m200 | 65.00 | 2000.00 | **48.00** | **53.60** | **57.00** | 1263.25 | 49.00 | 53.83 | 58.00 | 1505.69 | 2.24 | +1.00 |
| MDG-a_39_n2000_m200 | 60.00 | 2000.00 | 51.00 | 54.00 | **58.00** | 1234.13 | **48.00** | **53.48** | **58.00** | 1451.99 | 2.36 | -3.00 |
| MDG-a_40_n2000_m200 | 62.00 | 2000.00 | 50.00 | **53.20** | **56.00** | 1056.70 | **49.00** | 54.03 | 59.00 | 1361.78 | 2.30 | -1.00 |
| Average value (wins) | 38.08 | 1087.86 | 30.84(7) | 33.04(10) | **35.17**(19) | 709.86 | **29.92**(33) | **32.86**(30) | 35.49(21) | 849.34 | 1.32 | -0.92 |

Table 7. Comparative results of the proposed ILS_MinDiff algorithm and reference algorithms on *MDG-b*.

| | GRASP_EPR [13](2015) | | VNS_MinDiff [28](2016) | | | | ILS_MinDiff | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | $f_{best}$ | $t_{best}$ | $f_{best}$ | $f_{avg}$ | $f_{worst}$ | $t_{avg}$ | $f_{best}$ | $f_{avg}$ | $f_{worst}$ | $t_{avg}$ | $\sigma$ | $\Delta f_{best}$ |
| MDG-b_1_n500_m50 | 1350.08 | 178.54 | 1185.11 | 1246.78 | 1296.49 | 266.17 | **1118.48** | **1209.70** | **1265.10** | 198.74 | 35.72 | -66.63 |
| MDG-b_2_n500_m50 | 1368.54 | 189.36 | 1182.48 | 1256.77 | 1322.03 | 245.14 | **1082.03** | **1199.86** | **1249.69** | 279.30 | 35.52 | -100.45 |
| MDG-b_3_n500_m50 | 1286.01 | 186.81 | **1070.87** | 1243.84 | 1310.09 | 331.21 | 1104.07 | **1203.60** | **1265.66** | 237.08 | 39.72 | +33.20 |
| MDG-b_4_n500_m50 | 1300.24 | 185.34 | 1153.93 | 1240.57 | 1287.46 | 239.95 | **1052.27** | **1193.79** | **1258.01** | 226.04 | 43.67 | -101.66 |
| MDG-b_5_n500_m50 | 1258.79 | 185.03 | 1209.80 | 1262.90 | 1317.82 | 186.06 | **1051.91** | **1204.83** | **1275.55** | 225.86 | 48.06 | -157.89 |
| MDG-b_6_n500_m50 | 1272.73 | 182.13 | 1071.61 | 1227.71 | 1319.86 | 298.82 | **1061.50** | **1202.49** | **1292.94** | 242.20 | 41.70 | -10.11 |
| MDG-b_7_n500_m50 | 1279.10 | 193.63 | 1099.68 | 1215.38 | 1311.55 | 256.32 | **1076.57** | **1205.87** | **1260.31** | 229.67 | 38.66 | -23.11 |
| MDG-b_8_n500_m50 | 1315.79 | 185.12 | 1185.59 | 1245.45 | 1316.97 | 247.01 | **1005.45** | **1207.23** | **1278.51** | 250.86 | 47.25 | -180.14 |
| MDG-b_9_n500_m50 | 1346.91 | 175.09 | 1154.33 | 1232.61 | 1261.83 | 243.90 | **1116.63** | **1210.19** | **1260.14** | 269.01 | 35.30 | -37.70 |
| MDG-b_10_n500_m50 | 1339.82 | 179.28 | 1198.08 | 1242.15 | 1289.55 | 272.05 | **1092.25** | **1195.55** | **1264.05** | 237.71 | 44.78 | -105.83 |
| MDG-b_11_n500_m50 | 1305.28 | 182.65 | 1145.73 | 1221.54 | 1275.68 | 249.64 | **1090.59** | **1202.24** | **1256.12** | 236.51 | 36.85 | -55.14 |
| MDG-b_12_n500_m50 | 1274.36 | 169.72 | 1165.43 | 1238.15 | 1294.60 | 252.95 | **1093.89** | **1206.29** | **1287.75** | 232.29 | 39.64 | -71.54 |
| MDG-b_13_n500_m50 | 1337.33 | 185.02 | 1180.43 | 1238.00 | **188.52** | 130.92 | **1123.26** | **1205.87** | 1296.97 | 268.81 | 37.48 | -57.17 |
| MDG-b_14_n500_m50 | 1291.06 | 191.77 | 1166.81 | 1247.25 | 1315.79 | 150.65 | **1089.38** | **1200.78** | **1258.22** | 264.85 | 36.19 | -77.43 |
| MDG-b_15_n500_m50 | 1278.86 | 186.00 | 1220.83 | 1273.74 | 1314.10 | 281.51 | **1108.10** | **1207.73** | **1287.25** | 231.45 | 41.54 | -112.73 |
| MDG-b_16_n500_m50 | 1328.66 | 180.79 | 1176.16 | 1248.31 | 1317.30 | 295.13 | **1132.34** | **1217.24** | **1266.44** | 313.70 | 31.32 | -43.82 |
| MDG-b_17_n500_m50 | 1299.00 | 179.15 | 1174.66 | 1252.52 | 1297.05 | 319.81 | **1062.69** | **1200.76** | **1275.49** | 228.43 | 44.04 | -111.97 |
| MDG-b_18_n500_m50 | 1321.87 | 174.22 | 1187.82 | 1267.94 | 1338.04 | 152.27 | **1032.54** | **1198.20** | **1271.48** | 253.42 | 45.10 | -155.28 |
| MDG-b_19_n500_m50 | 1333.22 | 172.76 | 1175.26 | 1257.81 | 1291.88 | 369.35 | **1076.27** | **1206.94** | **1272.92** | 269.22 | 39.22 | -98.99 |
| MDG-b_20_n500_m50 | 1328.53 | 172.66 | 1151.34 | 1233.04 | 1285.53 | 271.70 | **1050.33** | **1199.99** | **1260.42** | 242.06 | 42.23 | -101.01 |
| MDG-b_21_n2000_m200 | 5073.98 | 2000.00 | 4083.16 | 4468.62 | 4737.59 | 1025.80 | **3978.52** | **4299.38** | **4552.68** | 1366.04 | 138.11 | -104.64 |
| MDG-b_22_n2000_m200 | 5062.07 | 2000.00 | 4187.77 | 4540.42 | 4952.63 | 1039.80 | **3911.34** | **4377.97** | **4678.26** | 1466.30 | 178.38 | -276.43 |
| MDG-b_23_n2000_m200 | 4899.35 | 2000.00 | 4237.38 | 4489.07 | 5171.39 | 1327.39 | **4127.34** | **4422.12** | **4858.86** | 1591.88 | 183.29 | -110.04 |
| MDG-b_24_n2000_m200 | 4780.51 | 2000.00 | 4212.28 | 4452.33 | 4708.87 | 1002.08 | **4088.26** | **4421.77** | **4705.92** | 1513.38 | 147.62 | -124.02 |
| MDG-b_25_n2000_m200 | 5021.93 | 2000.00 | 4152.88 | 4435.36 | 4713.25 | 1375.81 | **3892.67** | **4340.78** | **4577.78** | 1463.02 | 132.66 | -260.21 |
| MDG-b_26_n2000_m200 | 4959.65 | 2000.00 | **4039.92** | 4497.39 | 4798.83 | 1317.73 | 4116.90 | **4423.07** | **4775.00** | 1535.76 | 147.65 | +76.98 |
| MDG-b_27_n2000_m200 | 4874.36 | 2000.00 | **4010.77** | 4486.90 | 4855.86 | 1079.71 | 4126.90 | **4424.59** | **4711.53** | 1599.67 | 150.02 | +116.13 |
| MDG-b_28_n2000_m200 | 5245.69 | 2000.00 | 4206.07 | 4498.25 | **4798.32** | 844.44 | **4112.43** | **4446.16** | 4975.79 | 1579.61 | 173.71 | -93.64 |
| MDG-b_29_n2000_m200 | 4955.58 | 2000.00 | 4214.79 | 4505.51 | **4809.00** | 1037.28 | **4057.62** | **4377.08** | 4846.05 | 1450.96 | 164.65 | -157.17 |
| MDG-b_30_n2000_m200 | 5045.63 | 2000.00 | 4272.07 | 4564.38 | **4786.12** | 1022.86 | **4110.61** | **4470.64** | 4885.52 | 1519.61 | 178.31 | -161.47 |
| MDG-b_31_n2000_m200 | 4962.72 | 2000.00 | 4328.97 | 4474.43 | 4710.96 | 1248.66 | **4074.80** | **4323.11** | **4539.19** | 1155.72 | 120.41 | -254.17 |
| MDG-b_32_n2000_m200 | 4833.29 | 2000.00 | 4226.55 | 4484.07 | 4664.58 | 1069.63 | **3929.49** | **4301.35** | **4628.75** | 1278.11 | 159.57 | -297.06 |
| MDG-b_33_n2000_m200 | 4973.32 | 2000.39 | 4037.50 | 4387.64 | 4786.52 | 1281.73 | **3985.32** | **4351.01** | **4624.97** | 1396.08 | 148.77 | -52.18 |
| MDG-b_34_n2000_m200 | 4880.74 | 2000.00 | 4279.58 | 4480.58 | **4850.85** | 1038.31 | **4084.46** | **4402.11** | 4965.12 | 1611.59 | 183.27 | -195.12 |
| MDG-b_35_n2000_m200 | 5061.54 | 2000.00 | 4018.60 | **4367.05** | **4679.23** | 1582.57 | **4000.31** | 4396.43 | 4824.11 | 1432.83 | 163.53 | -18.30 |
| MDG-b_36_n2000_m200 | 4963.93 | 2000.00 | 4231.38 | **4433.05** | **4674.14** | 1067.68 | **4095.13** | 4435.33 | 4955.35 | 1678.21 | 172.50 | -136.25 |
| MDG-b_37_n2000_m200 | 4801.03 | 2000.00 | 4100.54 | 4472.45 | **4834.64** | 1479.05 | **4035.74** | **4409.06** | 4873.27 | 1524.78 | 164.91 | -64.80 |
| MDG-b_38_n2000_m200 | 4946.67 | 2000.00 | 4136.67 | 4506.89 | **4802.26** | 1262.67 | **4126.69** | **4418.53** | 4817.94 | 1443.40 | 171.30 | -9.98 |
| MDG-b_39_n2000_m200 | 5095.33 | 2000.44 | 4242.30 | 4450.95 | **4635.06** | 1219.16 | **4131.87** | **4403.46** | 4807.67 | 1486.56 | 137.81 | -110.43 |
| MDG-b_40_n2000_m200 | 5001.89 | 2000.00 | 4249.76 | 4556.52 | 4804.78 | 1422.06 | **4306.02** | **4306.02** | **4549.61** | 1232.07 | 138.38 | -226.72 |
| Average value (wins) | 3141.39 | 1090.90 | 2668.12(3) | 2861.11(2) | 3045.47(10) | 720.12 | **2565.75**(37) | **2795.73**(38) | **3013.91**(30) | 856.57 | 98.97 | -102.37 |

algorithms by finding 19 improved best solutions. In terms of the average solution value and the worst solution value (only available for ILS_MinDiff and VNS_MinDiff), ILS_MinDiff achieves a competitive performance compared with VNS_MinDiff, by winning 11 and 9 out of 20 instances respectively. For the last 9 out out of 10 large instances, VNS_MinDiff has a better performance in terms of the average solution value and the worst solution value, but consumes much more time than ILS_MinDiff. However, according to the explanation given at the beginning of this section, the computer used to run VNS_MinDiff is about 1.2 times faster than our computer. To verify if ILS_MinDiff can improve its results on these 9 instances with a running time comparable to that used by VNS_MinDiff (i.e., $3000 \times 1.2 = 3600$ seconds), we re-ran ILS_MinDiff with this relaxed time limit of 3600 seconds. ILS_MinDiff actually achieved better results than VNS_MinDiff in terms of the best, average and worst solution values. For example, the best, average, and worst values of ILS_MinDiff for *MDG-c_11_n3000_m500* are respectively improved to 10295.00, 11123.25 and 12214.00 with an average time $t_{avg} = 3237.36$ seconds. Similarly, for *MDG-c_12_n3000_m500*, we obtained $f_{best} = 9909.00$, $f_{avg} = 10894.98$, $f_{worst} = 12328.00$ and $t_{avg} = 3132.11$.

Finally, Table 9 summarizes the performances of the proposed ILS_MinDiff algorithm against the best performing reference VNS_MinDiff algorithm on all 190 benchmark instances. The significant differences are marked in bold. From this table, we can make the following observations:

- ILS_MinDiff performs highly competitively compared to the state-of-the-art results, by wining 152.5, 154 and 128 out of 190 instances in terms of the best solution value, the average solution value and the worst solution value respectively.
- ILS_MinDiff performs significantly better than the current best algorithm (VNS_MinDiff) in terms of the *best* solution value for 4 out of 6 data sets (*SOM-b*, *MDG-a*, *MDG-b* and *MDG-c*). For *GKD-b* and *GKD-c*, ILS_MinDiff respectively wins 31.5 and 14 instances, which are just slightly smaller than the corresponding critical values (i.e., $CV_{0.05}^{50} = 32$ and $CV_{0.05}^{20} = 15$).
- Compared to the reference algorithm, ILS_MinDiff significantly performs better in terms of the *average* solution value for all data sets except for *MDG-c*. Although the difference between ILS_MinDiff and VNS_MinDiff is not significant on *MDG-c*, ILS_MinDiff still achieves a better average solution value for 11 out of 20 instances.
- ILS_MinDiff performs significantly better than VNS_MinDiff in terms of the *worst* solution value on *SOM-b*, *GKD-b*, *GKD-c* and *MDG-b*. For *MDG-a* and *MDG-c*, ILS_MinDiff achieves a competitive performance, but the difference is not statistically significant.

Table 8. Comparative results of the proposed ILS_MinDiff algorithm and reference algorithms on *MDG-c*.

| | GRASP_EPR [13](2015) | | VNS_MinDiff [28](2016) | | | | ILS_MinDiff | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | $f_{best}$ | $t_{best}$ | $f_{best}$ | $f_{avg}$ | $f_{worst}$ | $t_{avg}$ | $f_{best}$ | $f_{avg}$ | $f_{worst}$ | $t_{avg}$ | $\sigma$ | $\Delta f_{best}$ |
| MDG-c_1_n3000_m300 | 7429.00 | 3001.50 | 6344.00 | 6595.40 | 7135.00 | 1455.45 | **5772.00** | **6265.60** | **6794.00** | 2322.74 | 290.56 | -572.00 |
| MDG-c_2_n3000_m300 | 7781.00 | 3001.59 | 6109.00 | 6651.50 | 7183.00 | 1320.28 | **5936.00** | **6539.33** | **7114.00** | 2772.71 | 300.02 | -173.00 |
| MDG-c_3_n3000_m300 | 7438.00 | 3001.63 | 6365.00 | 6828.70 | 7221.00 | 1639.72 | **5585.00** | **6243.03** | **7006.00** | 2501.82 | 277.13 | -780.00 |
| MDG-c_4_n3000_m300 | 7212.00 | 3001.71 | 6304.00 | 6787.10 | 7215.00 | 1294.78 | **5969.00** | **6636.75** | **7168.00** | 2841.41 | 299.52 | -335.00 |
| MDG-c_5_n3000_m300 | 7346.00 | 3001.48 | 5954.00 | 6729.30 | **7282.00** | 1648.19 | **5750.00** | **6663.25** | 7405.00 | 2750.35 | 343.32 | -204.00 |
| MDG-c_6_n3000_m400 | 10559.00 | 3002.86 | 8403.00 | 9422.10 | 10592.00 | 1861.19 | **7648.00** | **8412.98** | **8999.00** | 2781.03 | 334.32 | -755.00 |
| MDG-c_7_n3000_m400 | 9738.00 | 3003.16 | 8606.00 | 9308.60 | 9770.00 | 1847.39 | **7829.00** | **8457.15** | **9522.00** | 2837.35 | 373.83 | -777.00 |
| MDG-c_8_n3000_m400 | 10262.00 | 3002.85 | 8217.00 | 9206.80 | 10219.00 | 2009.81 | **7984.00** | **8497.28** | **9033.00** | 2699.22 | 240.93 | -233.00 |
| MDG-c_9_n3000_m400 | 10202.00 | 3003.00 | 8478.00 | 9140.50 | 10337.00 | 2082.95 | **7657.00** | **8259.35** | **9128.00** | 2626.88 | 371.85 | -821.00 |
| MDG-c_10_n3000_m400 | 9266.00 | 3003.02 | 8244.00 | 9372.30 | **10129.00** | 1821.70 | **7672.00** | **8646.00** | 10432.00 | 2992.54 | 589.01 | -572.00 |
| MDG-c_11_n3000_m500 | 13203.00 | 3005.46 | 11145.00 | **11998.90** | **13151.00** | 3034.47 | **11031.00** | 12223.38 | 14281.00 | 2840.16 | 679.63 | -114.00 |
| MDG-c_12_n3000_m500 | 13458.00 | 3005.06 | 11366.00 | **12001.40** | **12709.00** | 3138.85 | **10604.00** | 12103.03 | 13987.00 | 2832.65 | 782.51 | -762.00 |
| MDG-c_13_n3000_m500 | 11930.00 | 3004.86 | 10942.00 | **11832.40** | **12427.00** | 3322.91 | **10743.00** | 12228.58 | 14838.00 | 2867.29 | 949.06 | -199.00 |
| MDG-c_14_n3000_m500 | 13734.00 | 3005.04 | 10903.00 | **11455.20** | **12095.00** | 2736.52 | **9941.00** | 11643.90 | 15043.00 | 2833.35 | 1207.82 | -962.00 |
| MDG-c_15_n3000_m500 | 12091.00 | 3004.80 | 11051.00 | **12311.90** | **13282.00** | 3255.82 | **10870.00** | 12365.85 | 14298.00 | 2883.88 | 757.94 | -181.00 |
| MDG-c_16_n3000_m600 | 16682.00 | 3007.55 | 13934.00 | **14732.10** | **15278.00** | 3446.15 | **13910.00** | 15801.65 | 18285.00 | 2858.17 | 1065.75 | -24.00 |
| MDG-c_17_n3000_m600 | 16673.00 | 3007.45 | 14086.00 | **14882.70** | **16184.00** | 3549.47 | **13676.00** | 15284.10 | 17002.00 | 2899.82 | 712.02 | -410.00 |
| MDG-c_18_n3000_m600 | 15307.00 | 3007.09 | **13415.00** | **14515.20** | **15385.00** | 3536.68 | 14011.00 | 15547.08 | 16893.00 | 2925.46 | 589.07 | +596.00 |
| MDG-c_19_n3000_m600 | 14812.00 | 3007.68 | 13850.00 | **14821.90** | **15976.00** | 3905.60 | **13538.00** | 15526.85 | 16729.00 | 2890.80 | 753.32 | -312.00 |
| MDG-c_20_n3000_m600 | 14462.00 | 3007.16 | 13532.00 | 14651.80 | 15396.00 | 3653.64 | **12415.00** | **13545.33** | **14697.00** | 2879.54 | 709.70 | -1117.00 |
| Average value (wins) | 11479.25 | 3004.25 | 9862.40(1) | 10662.29(9) | **11448.30**(11) | 2528.08 | **9427.05**(19) | **10544.52**(11) | 11932.70(9) | 2791.86 | 581.37 | -435.35 |

Table 9
A summary of win statistics between the proposed ILS_MinDiff algorithm (left part of each column) and the reference VNS_MinDiff algorithm (right part of each column) on all six data sets.

| Indicator | SOM-b | GKD-b | GKD-c | MDG-a | MDG-b | MDG-c |
|---|---|---|---|---|---|---|
| $f_{best}$ | **18** \| 2 | 31.5 \| 18.5 | 14 \| 6 | **33** \| 7 | **37** \| 3 | **19** \| 1 |
| $f_{avg}$ | **19** \| 1 | **36** \| 14 | **20** \| 0 | **30** \| 10 | **38** \| 2 | 11 \| 9 |
| $f_{worst}$ | **16** \| 4 | **32** \| 18 | **20** \| 0 | 21 \| 19 | **30** \| 10 | 9 \| 11 |

## 5 Experimental analysis

Now, we study the role of the parameters of ILS_MinDiff: the search depth $nbr_{max}$, the weak perturbation strength $p_w$ and the strong perturbation coefficient $\alpha$. The analysis was based on a set of 9 instances selected from all six data sets, i.e., *SOM-b_8_n200_m80*, *SOM-b_20_n500_m200*, *GKD-b_50_n150_m45*, *GKD-c_20_n500_m50*, *MDG-a_20_n500_m50*, *MDG-a_40_n2000_m200*, *MDG-b_20_n500_m50*, *MDG-b_40_n2000_m200* and *MDG-c_20_n3000_m600*. For each instance with $n < 2000$, we solved it 20 times, otherwise we solved it 10 times.

### *5.1 Effect of the search depth $nbr_{max}$*

To investigate the effect of the search depth $nbr_{max}$ on the performance of ILS_MinDiff, we first fixed the weak perturbation strength $p_w$ to 3 and the strong perturbation coefficient $\alpha$ to 1.0, i.e., $p_s = 1.0 \times n/m$, and then varied $nbr_{max}$ from 1.0 to 10.0 with a step size of 1. Figure 2 shows the behavior of ILS_MinDiff relative to $nbr_{max}$, where the X-axis indicates the values of $nbr_{max}$ while the Y-axis shows the best/average objective values.

Figure 2 reveals that the average solution values of ILS_MinDiff continuously decrease when the values of $nbr_{max}$ increase from 1.0 to 10.0 for all tested instances except for *MDG-c_20_n3000_m600*. For *MDG-c_20_n3000_m600*, the curve shows a large variation and it also decreases when $nbr_{max}$ increases to 5. We also observe that ILS_MinDiff obtains a superior best solution value when $nbr_{max}$ is 4 or 5 on all tested instances. This justifies the adopted setting ($nbr_{max} = 5$) shown in Table 2.

### *5.2 Effect of the weak perturbation strength $p_w$*

To study the effect of the weak perturbation strength $p_w$ on the performance of the proposed algorithm, we first fixed the search depth $nbr_{max}$ to 5 according

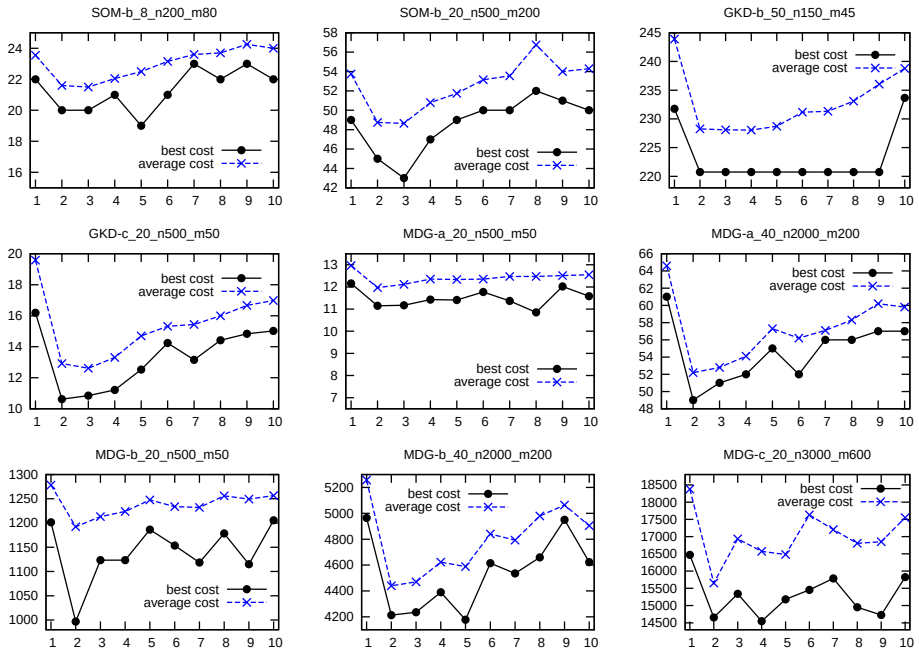


Fig. 3. Effect of the weak perturbation strength $p_w$ on the performance of the ILS_MinDiff algorithm

and the Y-axis show the values of $\alpha$ and the best/average objective values respectively.

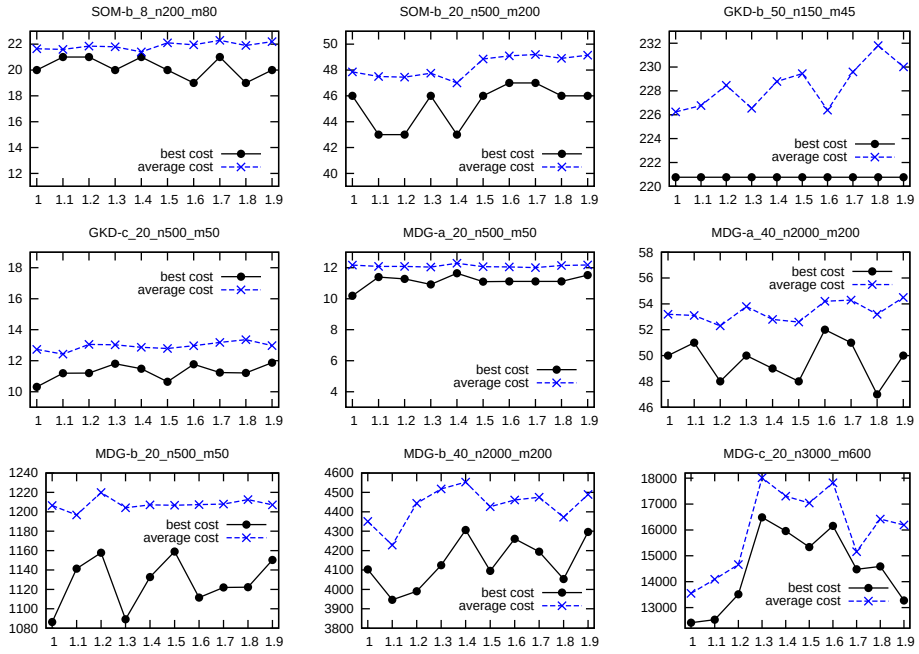


Fig. 4. Effect of the strong perturbation coefficient $\alpha$ on the performance of the ILS_MinDiff algorithm

As we observe from Figure 4, the average solution values are not really sensitive to $\alpha$ on almost all tested instances except for *MDG-c_20_n3000_m600*. To

achieve better average solution values, we roughly set $\alpha = 1.0$ for all instances.

## 6 Conclusions

The minimum differential dispersion problem (*Min-Diff DP*) is a useful model for a variety of practical applications. However, finding high-quality solutions to large *Min-Diff DP* instances represents an imposing computational challenge. In this study, we have proposed ILS_MinDiff, a highly effective iterated local search algorithm for *Min-Diff DP*, which adopts for the first time the general three-phase search framework to *Min-Diff DP*. The proposed algorithm uses the conventional swap neighborhood and integrates two specific features which distinguish itself from existing algorithms. It uses a local optima exploring procedure to locate nearby local optima within a limited search region and a local optima escaping procedure to displace the search to distant region. In addition, being conceptually simple, ILS_MinDiff was implemented with ease.

Extensive computational experiments on six data sets of 190 benchmark instances have demonstrated that despite its simplicity, the proposed algorithm competes very favorably with state-of-the-art methods in the literature. In particular, ILS_MinDiff is able to find new best results for 131 out of 190 instances (improved upper bounds) and match the best-known results for 42 out of the remaining 59 instances. These improved results constitute useful new references for evaluating other *Min-Diff DP* heuristic algorithms. The computational results indicate that the three-phase search framework is a very suitable for the studied problem when it is combined with the basic swap-based descent procedure and the local optima exploring and escaping procedures.

From this study, several perspectives can be contemplated for future studies. First, given its simplicity and effectiveness, the proposed algorithm can be advantageously employed as the local optimization component of a more complex hybrid algorithm (e.g., memetic search, evolutionary path-relinking). Second, this study demonstrates the effectiveness of the general iterated three-phase search framework for the minimum differential dispersion problem. It would be interesting to investigate the usefulness of this framework to solve other diversity problems like those presented in the introduction. Third, a more fundamental perspective concerns studies on a deep understanding of the proposed algorithm, which could provide useful information for additional improvements of the algorithm and the general search framework.

## Acknowledgment

We are grateful to the reviewers and Editor for their useful comments and suggestions which helped us to improve the paper. Support from the China Scholarship Council (2014-2018) for the first author of this paper is also acknowledged.

## References

[1] Aringhieri, R.; Cordone, R. & Grosso, A. Construction and improvement algorithms for dispersion problems, European Journal of Operational Research, 2015, 242, 21-33

[2] Benlic, U. & Hao, J.-K. A study of Breakout local search for the minimum sum coloring problem, 9th International Conference on Simulated Evolution and Learning, 2012, Lecture Notes in Computer Science, 7673, 128-137

[3] Benlic, U. & Hao, J.-K. Breakout local search for the quadratic assignment problem, Applied Mathematics and Computation, 2013, 219, 4800-4815

[4] Benlic, U. & Hao, J.-K. Breakout local search for maximum clique problems, Computers & Operations Research, 2013, 40, 192-206

[5] Benlic, U. & Hao, J.-K. Breakout local search for the max-cut problem, Engineering Applications of Artificial Intelligence, 2013, 26, 1162-1173

[6] Benlic, U. & Hao, J.-K. Breakout local search for the vertex separator problem, Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI), 2013, 461-467

[7] Brown, J. R. The sharing problem, Operations Research, INFORMS, 1979, 27, 324-340

[8] Brown, J. R. The knapsack sharing problem, Operations Research, INFORMS, 1979, 27, 341-355

[9] Carrasco, R.; Pham, A.; Gallego, M.; Gortázar, F.; Martí, R. & Duarte, A. Tabu search for the max-mean dispersion problem, Knowledge-Based Systems, 2015, 85, 256-264

[10] Demsar, J. Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research, 2006, 7, 1-30

[11] Ding, J.; Lü Z. P.; Cheng, T.C.E.; Xu, L. P. Breakout dynasearch for the single-machine total weighted tardiness problem, Computers & Industrial Engineering, 2016, 98, 1-10

[12] Duarte, A. & Martí, R. Tabu search and GRASP for the maximum diversity problem, European Journal of Operational Research, 2007, 178, 71-84

[13] Duarte, A.; Sánchez-Oro, J.; Resende, M. G.; Glover, F. & Martí, R. Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization, Information Sciences, 2015, 296, 46-60

[14] Fu, Z.-H. & Hao, J.-K. Breakout local search for the Steiner tree problem with revenue, budget and hop constraints, European Journal of Operational Research, 2014, 232, 209-220

[15] Fu, Z.-H. & Hao, J.-K. A three-phase search approach for the quadratic minimum spanning tree problem, Engineering Applications of Artificial Intelligence, 2015, 46, 113-130

[16] Ghandi, S. & Masehian, E. A breakout local search (BLS) method for solving the assembly sequence planning problem, Engineering Applications of Artificial Intelligence, 2015, 39, 245-266

[17] Ghosh, J. B. Computational aspects of the maximum diversity problem, Operations Research Letters, 1996, 19, 175-181

[18] Glover, F.; Kuo, C.-C. & Dhir, K. S. Heuristic algorithms for the maximum diversity problem, Journal of Information and Optimization Sciences, 1998, 19, 109-132

[19] de Kerchove, C. & Dooren, P. V. The PageTrust algorithm: How to rank web pages when negative links are allowed? Proceedings of the SIAM International Conference on Data Mining (SDM), 2008, 346-352

[20] Kortsarz, G. & Peleg, D. On choosing a dense subgraph, Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on, 1993, 692-701

[21] Kuby, M. J. Programming models for facility dispersion: The p-dispersion and maxisum dispersion problems, Geographical Analysis, 1987, 19, 315-329

[22] Lai, X. & Hao, J.-K. Iterated maxima search for the maximally diverse grouping problem, European Journal of Operational Research, 2016, 254, 780-800

[23] Lai, X. & Hao, J.-K. Iterated variable neighborhood search for the capacitated clustering problem, Engineering Applications of Artificial Intelligence, 2016, 56, 102-120

[24] Loureno, H.R.; Martin, O.C. & Stützle, T. Iterated local search, Handbook of Metaheuristics, 2003, 320-353

[25] Ma, F. & Hao, J.-K. A multiple search operator heuristic for the max-k-cut problem, Annals of Operations Research, 2016, 1-39

[26] Martí, R.; Gallego, M. & Duarte, A. A branch and bound algorithm for the maximum diversity problem, European Journal of Operational Research, 2010, 200, 36-44

[27] Martí, R.; Gallego, M.; Duarte, A. & Pardo, E. G. Heuristics and metaheuristics for the maximum diversity problem, Journal of Heuristics, 2013, 19, 591-615

[28] Mladenović, N.; Todosijević, R. & Urošević, D. Less is more: basic variable neighborhood search for minimum differential dispersion problem, Information Sciences, 2016, 326, 160-171

[29] Palubeckis, G. Iterated tabu search for the maximum diversity problem, Applied Mathematics and Computation, 2007, 189, 371-383

[30] Prokopyev, O.A.; Kong, N. & Martinez-Torres, D.L. The equitable dispersion problem, European Journal of Operational Research, 2009, 197, 59-67

[31] Silva, G.C.; Ochi, L.S. & Martins, S. L. Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem, International Workshop on Experimental and Efficient Algorithms, 2004, 498-512

[32] Yang, B.; Cheung, W. & Liu, J. Community mining from signed social networks, IEEE Transactions on Knowledge and Data Engineering, IEEE, 2007, 19, 1333-1348

[33] Yang, B.; Chen, H.; Zhao, X; Naka, M. & Huang J. On characterizing and computing the diversity of hyperlinks for anti-spamming page ranking, Knowledge-Based Systems, 2015, 77, 56-67

[34] Zhou, Y.; Hao, J.-K. & Goëffon, A. A three-phased local search approach for the clique partitioning problem, Journal of Combinatorial Optimization, 2016, 32(2), 469-491