Intensification-driven tabu search for the minimum differential dispersion problem

Xiangjing Lai [a], Jin-Kao Hao [b,c,*], Fred Glover [d], Dong Yue [a]

[a] *Institute of Advanced Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*

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

[c] *Institut Universitaire de France, 1 Rue Descartes, 75231 Paris, France*

[d] *OptTek Systems, Inc., 2241 17th Street Boulder, Colorado 80302, USA*

Knowledge-Based Systems, 5 January 2019

Abstract

The minimum differential dispersion problem is a NP-hard combinatorial optimization problem with numerous relevant applications. In this paper, we propose an intensification-driven tabu search algorithm for solving this computationally challenging problem by integrating a constrained neighborhood, a solution-based tabu strategy, and an intensified search mechanism to create a search that effectively exploits the elements of intensification and diversification. We demonstrate the competitiveness of the proposed algorithm by presenting improved new best solutions for 127 out of 250 benchmark instances (> 50%). We study the search trajectory of the algorithm to shed light on its behavior and investigate the spatial distribution of high-quality solutions in the search space to motivate the design choice of the intensified search mechanism.

Keywords: Combinatorial optimization; Dispersion problem; Tabu search; Candidate list strategy; Intensification mechanism; Heuristics.

---

1 Introduction

Dispersion problems are an important class of subset selection problems in binary optimization that have recently received substantial attention from the

* Corresponding author.

Email addresses: laixiangjing@gmail.com (Xiangjing Lai), jin-kao.hao@univ-angers.fr (Jin-Kao Hao), glover@opttek.com (Fred Glover), medongy@vip.163.com (Dong Yue).

combinatorial optimization community for their extensive practical applications. Dispersion problems can be roughly described as follows. Given a set $N = \{1, 2, \ldots, n\}$ of n elements and a distance matrix $[d_{ij}]_{n\times n}$ ($d_{ij} \geq 0$) defined on these elements, a dispersion problem is to select a subset M from N to optimize an objective f over the elements of M.

By varying the optimization objective, a variety of dispersion problems have been introduced and investigated in the literature, including notably the maximum diversity problem (MDP) [2,16,29,32], the max–min diversity problem (Max-Min DP) [11,24,26], the minimum differential dispersion problem (Min-Diff DP) [3,13,22,27,33], the maximum min-sum dispersion problem (Max-Minsum DP) [1,19,21,25], and the maximum mean dispersion problem (MaxMean DP) [6,12,17]. While MDP and Max-Min DP focus only on the dispersion criterion of the selected elements, Min-Diff DP, Max-Minsum DP, and MaxMean DP additionally consider the dispersion equity of solutions.

Practical application of dispersion problems covers a wide range, as represented by the problems of maximally diverse or similar group selection [1], urban public facility location [4], densest k-subgraph identification [5], equity-based measures in network flows [7], selection of homogeneous groups [8], facility location [14], web page ranking [20], and community mining [31]. These dispersion problems are NP-hard in the general case [25], and thus it is unlikely that a polynomial time algorithm exists to solve them unless $P = NP$.

In this study, we focus on Min-Diff DP that is known to be particularly difficult from a computational point of view [25]. Specifically, Min-Diff DP can be described as follows. Given a set $N = \{1, 2, \ldots, n\}$, an associated distance matrix $[d_{ij}]_{n\times n}$ ($d_{ij} \geq 0$ for $i \neq j$; $d_{ii} = 0$ for $\forall i$), and a fixed positive integer m, Min-Diff DP involves selecting a subset M of exactly m elements from N, such that the difference between the maximum sum and minimum sum of distances between a selected element and other selected elements in M is minimized. Formally, the Min-Diff DP problem can be written as:

$$\text{Minimize} \quad Max_{i\in M}\{\sum_{j\in M} d_{ij}\} - Min_{i\in M}\{\sum_{j\in M} d_{ij}\} \tag{1}$$

$$\text{Subject to} \quad M \subset N, |M| = m \tag{2}$$

Due to its strongly NP-hard character and its potential applications, Min-Diff DP has received particular attention within the general class of dispersion problems and has been the subject of a variety of solution approaches. In 2009, Prokopyev et al. [25] proposed a linear 0–1 mixed integer programming (MIP) formulation for Min-Diff DP and solved a number of small instances with $n \leq 100$ by means of the CPLEX 9.0 solver. Their computational results showed that the CPLEX solver used in these tests is very time-consuming even for small instances with $n = 50$. For example, for the instances with $n = 50$ and $m = 15$, the CPLEX 9.0 solver failed to obtain the optimal solution under a time limit of one hour. More modern versions of CPLEX run faster based on exploiting multiple cores, but without this boost the run times are very similar. Thus, for larger instances, heuristic algorithms are more appropriate to obtain near-optimal solutions and noteworthy advances have been made in just the past few years.

In 2015, Aringhieri et al. introduced a construction and improvement heuristic (CIH) algorithm for solving Min-Diff DP, which is composed of an initial solution construction stage and an improvement stage [3]. In the same year, Duarte et al. proposed a sophisticated evolutionary path relinking (EPR) algorithm by integrating a GRASP procedure, a variable neighborhood search (VNS) procedure, and an exterior path relinking operator [13]. Their computational results show that the EPR algorithm outperforms the basic GRASP algorithm in [25]. In 2016, based on the popular swap neighborhood, Mladenović et al. presented a basic VNS algorithm [22], and performed the experimental tests showing that this algorithm significantly outperformed the previous EPR algorithm. Recently (2017), Zhou et al. proposed an iterated local search (ILS) algorithm [33], which improved the best known results for a number of instances commonly used in the literature. Very recently (2017), Wang et al. devised a solution-based tabu search algorithm and a memetic algorithm [27], showing that their tabu search algorithm improved 71% of the previous best results and the memetic algorithm (which contained an embedded tabu search algorithm) improved 62% of the previous best results. This naturally raises the question of whether some combination of metaheuristics strategies may make it possible to do still better.

Recent studies show that solution-based tabu search [9,10,30] is more effective than the traditional attribute-based tabu search [15] for solving certain classes of binary optimization problems [27]. As reported in [27], the solution-based tabu search has been especially effective for Min-Diff DP. In this work, we go a step further by introducing an intensification-driven tabu search (IDTS) algorithm that extends the solution-based tabu search framework by integrating three special features: a new constrained swap neighborhood relying on a candidate list strategy, an enhanced tabu list management using three hash functions, and an intensified search mechanism to reinforce the search around high-quality solutions discovered. Computational results on 250 instances show that our IDTS algorithm is very competitive compared to the state-of-the-art algorithms in the literature, improving more than half of the currently best known solutions (127 out of 250 instances) while consuming a short computational time.

The remainder of the paper is organized as follows. Section 2 describes our IDTS algorithm in greater detail. In Section 3, we assess its performance in a computational study of 250 benchmark instances commonly used in the literature and provide a direct comparison with state-of-the-art algorithms for this problem. In Section 4, we discuss essential components of the IDTS algorithm and study their influence on its behavior. Section 5, which concludes the paper, summarizes the present work and provides research perspectives for future work.

2 Intensification-driven tabu search for Min-Diff DP

2.1 General Procedure

We elaborate the elements of the IDTS algorithm by means of the pseudo-code in Algorithm 1, where H_1, H_2, H_3 represent hash vectors used to define three tabu lists of length L, and h_1, h_2, h_3 represent the hash functions used to determine the tabu status of neighbor solutions referenced by these vectors. Finally, s and s^* respectively denote the current solution and the best solution found so far.

The IDTS algorithm starts by initializing the hash vectors that serve as tabu lists (lines 1–3), and then generates a feasible initial solution (line 4). Next, the algorithm enters a loop to execute the intensified search step (line 7), incorporating an inner 'while' loop (lines 8–20), to improve the incumbent solution, and these loops are repeatedly performed until the timeout limit t_{max} is reached. Specifically, the inner 'while' loop iterates until the current solution cannot be improved during the last α consecutive iterations, where α is a parameter called the tabu search depth. At each execution of the 'while' loop, a best eligible neighbor solution $s^{'}$ satisfying $H_1(h_1(s^{'})) \wedge H_2(h_2(s^{'})) \wedge H_3(h_3(s^{'})) = 0$ (i.e., a best neighbor solution not forbidden by the tabu lists, as discussed in Section 2.5) is selected from the current neighborhood $N^{\theta}_{swap}(s)$ defined in the following Section 2.4 to replace the incumbent solution s, and then the hash vectors H_k $(k = 1, 2, 3)$ are accordingly updated by the new incumbent solution s (line 19). After each tabu search run (i.e., when the 'while' loop terminates), the process switches to the intensified search step (line 7) and starts the next tabu search run with the best solution recorded in s^* as its initial solution. Finally, the algorithm returns the best solution found during the search and stops when the given time limit t_{max} is reached.

The intensified search step is one of key operations of the algorithm. As shown in previous studies [18,24], for a number of combinatorial optimization problems, high-quality solutions are not uniformly distributed in the search space. Instead, they are grouped in clusters, in accordance with the proximate optimality principle [15], where high-quality solutions at one level are hypothesized

Algorithm 1: General procedure of the intensification-driven tabu search (IDTS) algorithm for the Min-Diff DP problem

Input: Instance I, hash vectors H_1, H_2, H_3 with a length of L, hash functions h_1, h_2, h_3, parameter θ, cutoff time t_{max}, and tabu search depth α

Output: The best solution s^* found so far

`/* Initialization of hash vectors (tabu lists), Sect. 2.5 */`

for $i \leftarrow 0$ ***to*** $L-1$ **do**
  $H_1[i] \leftarrow 0$; $H_2[i] \leftarrow 0$; $H_3[i] \leftarrow 0$
end
$s \leftarrow InitialSolution(I)$ `/* Initial solution, Sect. 2.3 */`
$s^* \leftarrow s$
`/* Main search process */`
repeat
  $s \leftarrow s^*$ `/* Switch to the best solution found so far */`
  $counter \leftarrow 0$ `/* Counter for consecut. non-improv.` s^* `iter. */`
  while $counter \leq \alpha$ **do**
    Find a best neighbor solution $s^{'}$ in terms of f that satisfies $H_1(h_1(s^{'})) \wedge H_2(h_2(s^{'})) \wedge H_3(h_3(s^{'})) = 0$ in the neighborhood $N^{\theta}_{swap}(s)$
    `/* A solution` $s^{'}$ `with` $H_1(h_1(s^{'})) \wedge H_2(h_2(s^{'})) \wedge H_3(h_3(s^{'})) = 0$ `is identified as an eligible solution, Sections 2.4 and 2.5 */`
    $s \leftarrow s^{'}$ `/* Update the incumbent solution */`
    if $f(s) < f(s^*)$ **then**
      $s^* \leftarrow s$ `/* Update the best solution found so far */`
      $countor \leftarrow 0$
    end
    else
      $countor \leftarrow countor + 1$
    end
    `/* Update tabu lists, Sect. 2.5 */`
    $H_1[h_1(s)] \leftarrow 1$; $H_2[h_2(s)] \leftarrow 1$; $H_3[h_3(s)] \leftarrow 1$
  end
until $Time() \leq t_{max}$

to lie close to high-quality solutions at an adjacent level (defined relative to the moves employed or to a distance measure, depending on the case). These studies have demonstrated that high-quality solutions are typically found in the vicinity of other high-quality solutions by reference to the standard Euclidean distance measure. As we show in Section 4.5, this is also true for Min-Diff DP studied in this work. In such a circumstance, performing an intensified search around each newly discovered high-quality solution is clearly an advantageous strategy to find other high-quality solutions. The IDTS algorithm implements this strategy by using the intensified search step to enable the next tabu search run to systematically start its search from the best solution s^* found so far. Meanwhile, the tabu lists are not re-initialized after each intensified step and thus inherited by all tabu search runs. This ensures that each intensified search operation will lead to a different search trajectory even when the next tabu search run starts from the same starting point s^*. As a result, the nearby areas of s^* will be thoroughly examined and the intensification search of the algorithm is thus reinforced (Although different trajectories can also result by clearing or reducing the tabu search memory, in the present case we can continue to reap the benefits of the solution-based tabu strategy by retaining all previous memory).

2.2 Solution Representation, Search Space, and Evaluation Function

By reference to the set $N = \{1, 2, \ldots, n\}$, the distance matrix $[d_{ij}]_{n \times n}$, and the integer m, we can represent a subset $M \subset N$ by a n-dimensional binary vector $s = (x_1, x_2, \ldots, x_n)$, where $x_i = 1$ if the element i is selected to lie in M, and $x_i = 0$ otherwise. Equivalently, $s = (x_1, x_2, \ldots, x_n)$ can be indicated by a 2-tuple of sets (I^0, I^1) (i.e., $s = (I^0, I^1)$), where $I^0 = \{k : x_k = 0 \text{ in } s\}$ and $I^1 = \{k : x_k = 1 \text{ in } s\}$. An illustrative example for the solution representation is given in Fig. 1.

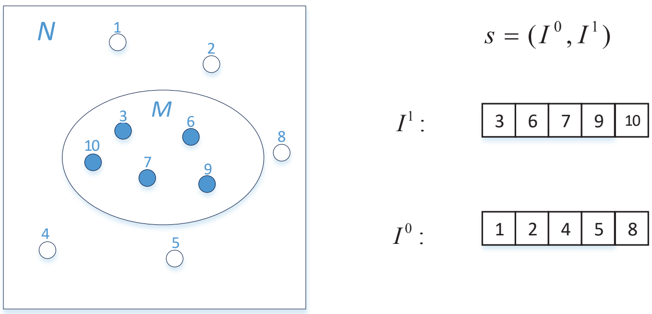

Fig. 1. An illustrative example for the solution representation, where the size of set N is 10 ($n = 10$) and the size of set M is 5 ($m = 5$).

The search space Ω_m explored by our IDTS algorithm is composed of all feasible solutions, i.e., $\Omega_m = \{(x_1, x_2, \ldots, x_n) : \sum_{i=1}^{i=n} x_i = m\}$, or equivalently, $\Omega_m = \{(I^0, I^1) : I^0, I^1 \subset N, |I^1| = m\}$. Obviously, the size of Ω_m is equal to $\frac{n!}{m!(n-m)!}$, which increases very quickly as the size of problem increases.

Given a solution $s = (I^0, I^1)$ in Ω_m, the objective function value $f(s)$ used to measure the quality of s is given by:

$$f(s) = Max_{i \in I^1}\{\sum_{j \in I^1} d_{ij}\} - Min_{i \in I^1}\{\sum_{j \in I^1} d_{ij}\} \tag{3}$$

Finally, for two solutions s_1 and s_2 in the search space, s_1 is better than s_2 if $f(s_1) < f(s_2)$ since f is to be minimized.

2.3 Initial Solution

Algorithm 2: Initial Solution Method

Function *InitialSolution()*
Input: $N = \{1, 2, \ldots, n\}$, m
Output: A feasible initial solution $s_0 = (x_1, x_2, \ldots, x_n)$
for $i \leftarrow 1$ ***to*** n **do**
    $x_i \leftarrow 0$
end
$c \leftarrow 0$
while $c < m$ **do**
    while *True* **do**
        $i \leftarrow rand()\ mod\ n$ `/* Randomly select a variable` x_i `*/`
        if $x_i = 0$ **then**
            break
        end
    end
    $x_i \leftarrow 1$
    $c \leftarrow c + 1$
end
return $(x_1, x_2, \ldots, x_n)$

The IDTS algorithm starts with an initial feasible solution s_0 generated by a randomized initialization procedure whose pseudo-code is given in Algorithm 2. The initialization procedure randomly selects m distinct variables x_i from $\{x_1, x_2, \ldots, x_n\}$ to be assigned the value of 1, while assigning the remaining $n-m$ variables the value of 0 to create the initial solution of the IDTS algorithm.

2.4 Neighborhood Structure and Its Evaluation Technique

The neighborhood explored by our IDTS algorithm is defined by the swap operator $Swap(\cdot,\cdot)$ that is commonly used in previous studies for Min-Diff DP [3,13,22,27,33]. Given a solution $s = (I^0, I^1)$ and two elements $u \in I^0$ and $v \in I^1$, the $Swap(u, v)$ operation exchanges the positions of the elements u and v to generate a neighbor solution of s that we denote by $s \oplus Swap(u, v)$. For a solution $s = (I^0, I^1)$, the largest possible neighborhood $N_{swap}^{full}(s)$ (i.e., the full swap neighborhood) induced by the swap operator is composed of all possible solutions that can be obtained by applying the swap operator to s, i.e., $N^{full}_{swap}(s) = \{s \oplus Swap(u, v) : u \in I^0, v \in I^1\}$. The size $m \times (n - m)$ of neighborhood $N^{full}_{swap}(s)$ becomes relatively large when m approaches to $n/2$ even for the medium-sized instances, making an algorithm that examines the full neighborhood very time-consuming. Furthermore, unlike other local search methods (e.g., the first improvement descent method or the simulated annealing method), a tabu search algorithm typically seeks a highest evaluation move at each iteration. When faced with a large neighborhood, tabu search therefore employs a candidate list strategy designed to create a set of high-quality moves that is much smaller than the full neighborhood. A variety of candidate list strategies are presented in [15] and variations incorporating their underlying principles are introduced in [28,29,32].

To focus on the most promising neighbor solutions and thus reduce the computational effort of the IDTS algorithm, we adopt a candidate list strategy based on a constrained swap neighborhood N^{θ}_{swap} for Min-Diff DP, using a parameter θ to control the neighborhood size. Specifically, given a solution $s = (I^0, I^1)$, the elements to be swapped in I^0 are limited to a high-quality subset $X \subset I^0$ in N^{θ}_{swap}, which constitutes an instance of a *successive filter* candidate list strategy in [15]. Given such a subset X of I^0, the neighborhood $N^{\theta}_{swap}(s)$ can be formally written as $N^{\theta}_{swap}(s) = \{s \oplus Swap(u, v) : u \in X \subset I^0, v \in I^1\}$. Hence, N^{θ}_{swap} has a size of $m \times |X|$. Another form of a successive filter candidate list strategy similarly extracts a subset of I^1 to further reduce the size of the neighborhood examined, with an increased risk of reducing the quality of the best move in the resulting neighborhood.

To identify the subset X and evaluate the neighborhood N^{θ}_{swap} efficiently, the IDTS algorithm maintains a n-dimensional vector $\Delta = (\Delta_1, \Delta_2, \ldots, \Delta_n)$, where $\Delta_i = \sum_{j \in I^1} d_{ij}$. Specifically, the subset X is constructed as follows. First, the value $\delta = |\Delta_i - \frac{(\Delta_{min} + \Delta_{max})}{2}|$ is calculated for each element $i \in I^0$, where $\Delta_{min} = Min_{i \in I^1}\{\Delta_i\}$ and $\Delta_{max} = Max_{i \in I^1}\{\Delta_i\}$. Then, the elements in I^0 are sorted in an ascending order by a quick-sort method according to their δ values, since those elements having a small $\delta(i)$ value are the most promising to minimize the objective function if they are selected in the solution. Finally, the first $Min\{n - m, \theta \times n\}$ elements are selected to form the subset X. An illustrative example for the neighborhood generation strategy is given in Fig. 2.

Given a solution $s = (I^0, I^1)$ and its Δ vector $(\Delta_1, \Delta_2, \ldots, \Delta_n)$, the objective value $f(s)$ $(= Max_{i \in I^1}\{\Delta_i\} - Min_{i \in I^1}\{\Delta_i\})$ can be calculated in $O(m)$ time as described in the previous studies [3,13]. Moreover, when a swap move $Swap(u, v)$ is performed from the current solution s, the vector $(\Delta_1, \Delta_2, \ldots, \Delta_n)$ can be updated in $O(n)$ time as follows:

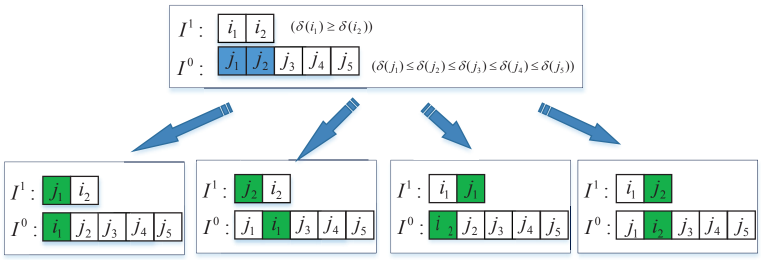

Fig. 2. An illustrative example for the neighborhood generation strategy, where the size of set N and the value of m are respectively 7 and 2, and the size of subset X is 2.

$$\Delta_i = \begin{cases} \Delta_i - d_{ui}, & \text{for } i = v; \quad (4) \\ \Delta_i + d_{vi}, & \text{for } i = u; \quad (5) \\ \Delta_i - d_{ui} + d_{vi}, & \text{otherwise}; \quad (6) \end{cases}$$

As such, the computational complexity of one iteration of our IDTS algorithm is bounded by $O(|X| \times m^2 + mlogm + (n-m)log(n-m) + n)$, where $mlogm + (n-m)log(n-m))$ is required by the quick-sort algorithm and represents a very small proportion of the total complexity.

Finally, the IDTS algorithm examines the neighborhood N^{θ}_{swap} in a lexicographical order and switches immediately to the next iteration as long as an improving solution is encountered. In this way, the algorithm can significantly be speeded up at the early stage of the algorithm.

2.5 Tabu List Management Strategy and Determination of Tabu Status

In the IDTS algorithm, we adopt the solution-based tabu strategy to determine the tabu status of neighbor solutions during the neighborhood evaluation. In principle, all solutions that have not been visited are considered as eligible solutions, while all the visited solutions are considered tabu and thus excluded from further consideration.

In our IDTS implementation, we adopt the technique of [19] and employ three hash vectors H_1, H_2, and H_3 (taking the role of the tabu lists) to determine the tabu status of neighbor solutions, where each hash vector H_k $(k = 1, 2, 3)$ is associated with a hash function h_k. Each hash vector H_k $(k = 1, 2, 3)$ is a L-dimensional binary vector (L is the length of the hash vectors), where $H_k[i]$ $(0 \le i \le L - 1)$ takes the value of 0 or 1. The hash functions h_k $(k = 1, 2, 3)$ are used to map the solutions of the search space Ω_m to the indices of the hash vectors H_k, i.e., $h_k : \Omega \rightarrow \{0, 1, 2, \ldots, L-1\}$ $(k = 1, 2, 3)$.

To be able to rapidly calculate the hash values of the neighbor solutions, we employ three simple hash functions inspired by the studies [9,27,30]. We define these three hash functions h_k $(k = 1, 2, 3)$ relative to a candidate solution $s = (x_1, x_2, \ldots, x_n)$ as follows:

$$h_k(s) = (\sum_{i=1}^{n} \lfloor i^{\xi_k} \rfloor \times x_i) \ mod \ L \tag{7}$$

where ξ_k $(k = 1, 2, 3)$ are parameters of the hash functions (see Section 3.2), while L is empirically set to 10^8.

In the IDTS algorithm, the hash vectors are maintained as follows. At the beginning, all hash vectors are initialized to 0 (lines 1–3 of Algorithm 1). Then, they are dynamically updated by the incumbent solution s as the search progresses, as shown in line 19 of Algorithm 1. Accompanying this, we calculate the hash values of neighbor solutions as follows. First, given the incumbent solution s and its hash value $h_k(s)$, the hash value of any neighbor solution $s^{'}(= s \oplus Swap(u, v))$ can be obtained in $O(1)$ by setting $h_k(s^{'})$ to $h_k(s) + (\lfloor v^{\xi_k} \rfloor - \lfloor u^{\xi_k} \rfloor)$. Second, for the initial solution s_{inital}, the hash value $h_k(s_{inital})$ must be calculated from scratch, and the associated time complexity is bounded by $O(n)$ for each hash function h_k $(k = 1, 2, 3)$ according to Eq.(7).

Using the three hash vectors defined above and the associated hash functions, the tabu status of neighbor solutions can be easily determined. A candidate neighbor solution $s^{'}$ is determined to be non-tabu if at least one of the three hash values $H_1[h_1(s^{'})]$, $H_2[h_2(s^{'})]$, and $H_3[h_3(s^{'})]$ is 0, since such a solution cannot have been visited. If instead all the hash values $H_1[h_1(s^{'})]$, $H_2[h_2(s^{'})]$, and $H_3[h_3(s^{'})]$ equal 1, then with high probability the neighbor solution $s^{'}$ has been visited previously and thus is considered as a tabu solution. In short, a neighbor solution $s^{'}$ is excluded from consideration if and only if $H_1(h_1(s^{'})) \wedge H_2(h_2(s^{'})) \wedge H_3(h_3(s^{'})) = 1$.

2.6 Relation with an Existing Tabu Search Algorithm

Our IDTS algorithm shares similarities with the tabu search algorithm of [27] in the sense that both algorithms are based on the general solution-based tabu approach. On the other hand, our IDTS algorithm has several features that distinguish it from the algorithm of [27]. The first is the parametric constrained swap neighborhood whose size is controlled by the parameter θ and which appreciably reduces the computational burden of our method. By contrast, the algorithm of [27] employs a randomized constrained neighborhood composed of solutions sampled according to a probability from the full swap neighborhood $N_{swap}^{full}(s)$, leading to a neighborhood of different size at each iteration of the algorithm. Second, to determine the tabu status of neighbor solutions, IDTS uses three hash vectors and the associated hash functions, instead of using two hash vectors as in [27], which considerably decreases the error rate of identifying the tabu status of a candidate solution. Third, our IDTS algorithm employs an intensified search mechanism, which is motivated by studying the distribution of high-quality solutions in the search space (see Section 4.5). Finally, as the experimental results in Section 4.3 demonstrate, our IDTS algorithm equipped with these features outperforms all existing methods including the latest tabu search algorithm and the memetic algorithm of [27].

3 Experimental Results and Comparisons

We assess the performance of the proposed IDTS algorithm by carrying out extensive computational experiments on a large number of commonly used benchmark instances. The computational results of the IDTS algorithm are provided and compared with those of the current leading algorithms in the literature.

3.1 Benchmark Instances

In the experiments, we employed eight sets of 250 benchmark instances [1] as our test bed. These instances have been widely used to assess algorithms for several dispersion problems, including the maximum diversity problem [32], Max-Minsum DP [1], and Min-Diff DP studied in this work [3,13,22,27,33]. The main characteristics of these benchmark instances are summarized as follows:

- APOM Set : 40 small instances with $n \in [50, 250]$ and $m \in \{0.2n, 0.4n\}$. Distances between elements are Euclidean or random integers in $[0, 10000]$.
- GKD-b set : 50 instances, where n varies from 25 to 150, m varies from 2 to 45, and distances are Euclidean.
- GKD-c Set : 20 instances with $n = 500$ and $m = 50$, and distances are Euclidean.
- SOM-b Set : 20 instances with $n \in [100, 500]$ and $m \in \{0.1n, 0.2n, 0.3n, 0.4n\}$, and distances are integers generated randomly in $[0, 9]$.
- DM1A Set : 20 instances with $n = 500$ and $m = 200$, and distances are a real number randomly generated in $[0, 10]$. These instances are renamed in

[1] Available at `http://www.di.unito.it/~aringhie/benchmarks.html` and `http://www.optsicom.es/mindiff/`

[27] as MDG-a_41 to MDG-a_60 .

- MDG-a Set : 20 instances with $n = 500$ and $m = 50$ and 20 instances with $n = 2000$ and $m = 200$. Like for DM1A, the distances are real numbers generated randomly in $[0, 10]$.
- MDG-b Set : 20 instances with $n = 500$ and $m = 50$ and 20 larger instances with $n = 2000$ and $m = 200$. The distances are real numbers generated randomly in $[0, 1000]$.
- MDG-c set : 20 large instances with $n = 3000$ and $m \in \{300, 400, 500, 600\}$, and distances are integers generated randomly in $[0, 1000]$.

3.2 Parameter Settings and Experimental Protocol

Table 1
Settings of parameters

Parameters	Section	Description	Values
α	2.4	depth of tabu search	{35,100}
θ	2.4	parameter used to construct the constrained neighborhood	{0.3,1.0}
ξ_1	2.5	parameter for the first hash function	1.8
ξ_2	2.5	parameter for the second hash function	1.9
ξ_3	2.5	parameter for the third hash function	2.0

The IDTS algorithm employs five parameters, whose values and descriptions are provided in Table 1. According to the parameter analysis in Section 4.1, the parameter θ used to control the neighborhood size was set to 0.3 except for the APOM and GKD-b instances for which θ was set to 1.0. The tabu search depth α was set to 35 except for the GKD-c instances for which it was set to 100. The parameters ξ_1, ξ_2, ξ_3 used to define the hash functions were respectively set to 1.8, 1.9, and 2.0.

To assess and compare the performance of the IDTS algorithm, we use the five most recent state-of-the-art Min-Diff DP algorithms in the literature as our main reference algorithms: the construction and improvement heuristic (CIH) [3], the evolutionary path relinking (EPR) algorithm [13], the variable neighborhood search (VNS) algorithm [22], the iterated local search (ILS) algorithm [33], and the solution-based tabu search (TS) algorithm [27]. Our IDTS algorithm and all the reference algorithms were implemented in the C++ programming language. and compiled using the g++ compiler with the -O3 flag as in [27,33]. For the CIH, EPR, VNS algorithms, the new versions implemented by the authors of [27] were used in our comparisons, since the new implementations of these algorithms have a much better performance than the original ones according to experimental results in [27]. Moreover, all the computational experiments and comparisons in this work are based on the same computing platform with an Xeon E5440 processor (2.83 GHz and 2G RAM), running the Linux operating system, which makes it possible to make a direct and fair comparison between the proposed IDTS algorithm and these reference algorithms.

Following the studies [13,22,33], our IDTS algorithm was run 20 times for each tested instance, with a time limit t_{max} equaling n seconds for each run, where n represents the number of elements in the tested instance.

3.3 Computational Results and Comparison

Our experimental results [2] are divided into two parts according to the recent studies [27,33], where the first part is based on 80 benchmark instances of four sets (DM1A, MDG-a with $n = 2000$, MDG-b with $n = 2000$, and MDG-c), and the second part includes the remaining 170 instances. In [27,33], all the tested algorithms were run on the same computing platform as our machine for the first part of experiments, which allows us to make a fair comparison between our IDTS algorithm and other algorithms by directly comparing our computational results with the results reported in [27,33]. However, for the remaining instances, the time limits were set according to special instances in reference [27], which makes a direct comparison between the algorithms difficult. For this reason, we focus in this section on the first part of experimental results, and provide our experimental results in the Appendix for the remaining instances, where we also report the previous best known results in the literature.

The computational results are summarized in Tables 2–9 respectively for benchmark sets DM1A, MDG-a with $n = 2000$, MDG-b with $n = 2000$, and MDG-c. The best results (f_{best}) over 20 independent runs are shown in Tables 2, 4, 6 and 8, and the average results (f_{avg}) are given in Tables 3, 5, 7, and 9. In Tables 2, 4, 6 and 8, the first three columns give the instance name, the time limit in seconds, and the previous best known objective value (f_{bkv}) in the literature (Best Known), and the last two columns indicate the best objective values obtained by our IDTS algorithm and the difference $\Delta_{fest}(= f_{best} - f_{bkv})$ between our best objective value and the previous best known objective value in the literature (A negative value indicates an improved best known result). For a few of instances the current best known results were only obtained by the combined memetic/tabu search algorithm of [27], although using a much longer time limit than that employed by our algorithm ($t_{max} = 20 \times n$ seconds, instead of $t_{max} = n$ seconds). Also, in a few instances no reference algorithm (i.e., no algorithm other than ours) was able to reach the previous best known result with the present time limit. Other columns give the best result obtained by the reference algorithms, including the CIH algorithm [3], the EPR algorithm [13], the VNS algorithm [22], the ILS algorithm [33], and the tabu search (TS) algorithm [27]. Similarly, in Tables 3, 5, 7, and 9, the first two

[2] Our solution certificates are available at: `http://www.info.univ-angers.fr/pub/hao/mindiffdp_IDTS.html`.

columns show the instance name and the time limit. The last two columns report the average objective values of our IDTS algorithm over 20 runs and the standard deviation (*std.*) of objective values, and other columns give the average objective values (f_{avg}) of the reference algorithms, respectively.

In addition, the row "Avg" in these tables shows the average value of each column, and the row "#Best" gives the number of instances for which an algorithm obtained the best results among the compared algorithms, where the previous best known results from the literature are also compared with f_{best} of the IDTS algorithm. To verify whether there exists a significant difference between the results of our IDTS algorithm and those of the reference algorithms, the *p-values* from the non-parametric Friedman tests are given in the last row of the tables, where a *p-value* less than 0.05 implies a significant difference between two groups of compared results. Finally, the best results among the compared results are indicated in bold in these tables, and the improved results (i.e., the new best known results) are marked by "*".

Table 2
Computational results and comparison in the best objective value obtained (f_{best}) on the DM1A instances.

			CIH [3]	EPR [13]	VNS [22]	TS [27]	IDTS (this work)	
Instance	Time (s)	Best known	f_{best}	f_{best}	f_{best}	f_{best}	f_{best}	Δf_{best}
01Type1_52.1_n500m200	500	**33.37**	41.29	55.26	49.15	36.49	34.77	1.40
02Type1_52.2_n500m200	500	**34.35**	42.80	56.03	50.69	38.72	34.60	0.25
03Type1_52.3_n500m200	500	**33.23**	41.88	53.44	47.64	38.34	34.71	1.48
04Type1_52.4_n500m200	500	**34.28**	41.22	53.23	46.85	38.60	34.94	0.66
05Type1_52.5_n500m200	500	35.02	42.28	54.84	47.19	38.18	**34.75***	-0.27
06Type1_52.6_n500m200	500	35.55	41.94	54.66	48.38	38.00	**33.97***	-1.58
07Type1_52.7_n500m200	500	35.41	41.42	54.87	47.15	37.34	**34.07***	-1.34
08Type1_52.8_n500m200	500	37.91	40.43	55.09	46.93	37.91	**34.00***	-3.91
09Type1_52.9_n500m200	500	**33.23**	41.08	53.82	47.59	38.68	34.01	0.78
10Type1_52.10_n500m200	500	**34.32**	41.66	54.18	46.29	38.03	34.84	0.52
11Type1_52.11_n500m200	500	36.48	42.93	56.78	48.74	38.07	**33.91***	-2.57
12Type1_52.12_n500m200	500	33.98	42.76	56.35	49.09	38.58	**33.73***	-0.25
13Type1_52.13_n500m200	500	35.84	42.58	57.07	47.88	38.77	**34.18***	-1.66
14Type1_52.14_n500m200	500	**33.20**	41.66	54.19	49.10	38.85	33.79	0.59
15Type1_52.15_n500m200	500	35.89	41.98	57.38	49.28	38.31	**35.58***	-0.31
16Type1_52.16_n500m200	500	**34.40**	41.72	54.45	48.10	39.19	35.16	0.76
17Type1_52.17_n500m200	500	38.28	40.67	52.11	48.75	38.50	**34.20***	-4.08
18Type1_52.18_n500m200	500	35.37	42.58	53.58	44.16	37.15	**34.18***	-1.19
19Type1_52.19_n500m200	500	36.46	41.18	54.06	45.83	38.91	**35.50***	-0.96
20Type1_52.20_n500m200	500	36.28	41.21	55.27	48.21	38.37	**35.22***	-1.06
Avg	500	35.14	41.76	54.83	47.85	38.25	**34.51**	-0.64
#Best		8	0	0	0	0	**12**	
p-value		3.71e-1	7.74e-6	7.74e-6	7.74e-6	7.74e-6		

Tables 2 and 3 for the set DM1A show that the IDTS algorithm performs much better in terms of f_{best} than the reference algorithms CIH, EPR, VNS, and TS. In particular, the IDTS algorithm yielded improved solutions for 12 out of 20 instances and obtained the best result in terms of "Avg" for all the cases. By contrast, none of the reference algorithms can attain the current best known results for these instances. Table 3 also shows that the IDTS

Table 3
Computational results and comparison in the average objective value obtained (f_{avg}) on the DM1A instances.

Instance	Time (s)	CIH [3]	EPR [13]	VNS [22]	TS [27]	IDTS (this work)	
		f_{avg}	f_{avg}	f_{avg}	f_{avg}	f_{avg}	*std.*
01Type1_52.1_n500m200	500	44.82	58.33	52.40	40.31	**37.98**	1.57
02Type1_52.2_n500m200	500	44.51	60.19	52.86	40.18	**37.99**	1.64
03Type1_52.3_n500m200	500	44.56	57.72	50.03	39.94	**37.46**	1.38
04Type1_52.4_n500m200	500	43.95	58.33	50.96	40.65	**38.14**	1.61
05Type1_52.5_n500m200	500	44.00	57.58	49.98	39.62	**37.29**	1.38
06Type1_52.6_n500m200	500	44.10	58.01	50.90	39.64	**38.57**	1.37
07Type1_52.7_n500m200	500	43.99	57.64	51.31	39.79	**38.02**	1.31
08Type1_52.8_n500m200	500	43.49	57.95	49.71	39.30	**37.21**	1.45
09Type1_52.9_n500m200	500	44.47	57.55	51.54	40.06	**37.60**	1.41
10Type1_52.10_n500m200	500	44.22	57.22	51.44	40.00	**37.47**	1.34
11Type1_52.11_n500m200	500	44.14	58.66	52.84	40.07	**37.83**	1.44
12Type1_52.12_n500m200	500	44.22	58.64	52.00	40.26	**37.95**	1.75
13Type1_52.13_n500m200	500	44.06	59.48	52.58	40.21	**37.87**	1.78
14Type1_52.14_n500m200	500	43.96	58.04	51.87	40.38	**36.96**	1.24
15Type1_52.15_n500m200	500	44.47	59.27	52.39	40.22	**38.03**	1.28
16Type1_52.16_n500m200	500	44.35	58.78	50.82	40.53	**37.90**	1.68
17Type1_52.17_n500m200	500	43.82	57.29	51.96	40.32	**37.90**	1.71
18Type1_52.18_n500m200	500	43.65	56.36	50.33	39.70	**37.42**	1.59
19Type1_52.19_n500m200	500	44.93	58.32	50.59	40.82	**38.50**	1.67
20Type1_52.20_n500m200	500	44.78	57.85	51.73	39.89	**37.98**	1.53
Avg.	500	44.22	58.16	51.41	40.09	**37.80**	1.51
#Best		0	0	0	0	**20**	
p-value		7.74e-06	7.74e-06	7.74e-06	7.74e-06		

Table 4
Computational results and comparison in the best objective value obtained (f_{best}) on the MDG-a instances with $n = 2000$.

Instance	Time (s)	Best known	CIH [3]	EPR [13]	VNS [22]	ILS [33]	TS [27]	IDTS (this work)	
			f_{best}	f_{best}	f_{best}	f_{best}	f_{best}	f_{best}	Δf_{best}
MDG-a_21_n2000_m200	2000	38	41	49	48	50	38	**34***	-4
MDG-a_22_n2000_m200	2000	37	40	51	49	50	37	**34***	-3
MDG-a_23_n2000_m200	2000	38	41	50	50	49	38	**34***	-4
MDG-a_24_n2000_m200	2000	38	42	49	50	50	39	**36***	-2
MDG-a_25_n2000_m200	2000	38	41	50	49	50	38	**34***	-4
MDG-a_26_n2000_m200	2000	38	40	48	47	50	38	**35***	-3
MDG-a_27_n2000_m200	2000	38	40	51	45	49	38	**34***	-4
MDG-a_28_n2000_m200	2000	38	41	47	47	50	38	**35***	-3
MDG-a_29_n2000_m200	2000	37	41	49	47	47	37	**34***	-3
MDG-a_30_n2000_m200	2000	38	38	51	45	49	38	**34***	-4
MDG-a_31_n2000_m200	2000	38	41	51	44	49	38	**35***	-3
MDG-a_32_n2000_m200	2000	38	40	50	46	48	38	**36***	-2
MDG-a_33_n2000_m200	2000	38	42	51	45	48	39	**35***	-3
MDG-a_34_n2000_m200	2000	38	41	49	50	49	38	**34***	-4
MDG-a_35_n2000_m200	2000	39	41	50	47	48	39	**36***	-3
MDG-a_36_n2000_m200	2000	37	41	50	51	48	38	**34***	-3
MDG-a_37_n2000_m200	2000	38	41	50	47	48	38	**34***	-4
MDG-a_38_n2000_m200	2000	38	41	52	47	49	38	**35***	-3
MDG-a_39_n2000_m200	2000	38	41	50	48	48	38	**34***	-4
MDG-a_40_n2000_m200	2000	37	41	50	48	49	37	**35***	-2
Avg.		37.85	40.75	49.9	47.5	48.9	38	**34.6**	-3.25
#Best		0	0	0	0	0	0	**20**	
p-value		7.74e-06	7.74e-06	7.74e-06	7.74e-06	7.74e-06	7.74e-06		

Table 5
Computational results and comparison in the average objective value obtained (f_{avg}) on the MDG-a instances with $n = 2000$.

Instance	Time (s)	CIH [3]	EPR [13]	VNS [22]	ILS [33]	TS [27]	IDTS (this work)	
		f_{avg}	f_{avg}	f_{avg}	f_{avg}	f_{avg}	f_{avg}	*std.*
MDG-a_21_n2000_m200	2000	43.30	53.80	50.40	53.43	39.45	**36.60**	1.24
MDG-a_22_n2000_m200	2000	42.20	54.15	50.85	53.55	39.25	**36.85**	1.19
MDG-a_23_n2000_m200	2000	43.45	53.70	52.70	53.60	40.05	**36.75**	1.58
MDG-a_24_n2000_m200	2000	43.15	54.05	53.10	53.63	39.65	**37.30**	0.78
MDG-a_25_n2000_m200	2000	42.55	54.80	52.85	53.60	39.45	**37.20**	1.25
MDG-a_26_n2000_m200	2000	42.15	54.00	50.10	53.58	39.95	**37.30**	1.35
MDG-a_27_n2000_m200	2000	42.20	55.15	49.40	53.73	40.30	**37.15**	1.96
MDG-a_28_n2000_m200	2000	42.50	56.05	50.40	52.98	39.50	**37.40**	1.36
MDG-a_29_n2000_m200	2000	42.40	53.05	50.30	53.48	39.15	**37.20**	1.21
MDG-a_30_n2000_m200	2000	42.30	54.85	50.85	54.28	39.50	**36.65**	1.06
MDG-a_31_n2000_m200	2000	42.65	54.25	49.40	53.88	39.50	**37.30**	1.05
MDG-a_32_n2000_m200	2000	42.45	54.15	49.10	53.25	39.60	**38.00**	1.22
MDG-a_33_n2000_m200	2000	43.10	53.90	49.35	53.80	40.35	**36.80**	1.25
MDG-a_34_n2000_m200	2000	42.50	55.20	52.60	53.48	39.50	**37.35**	1.46
MDG-a_35_n2000_m200	2000	42.10	55.75	50.35	54.08	40.35	**37.90**	1.09
MDG-a_36_n2000_m200	2000	42.60	53.70	52.60	53.73	39.40	**37.30**	1.31
MDG-a_37_n2000_m200	2000	42.65	54.90	49.35	53.85	39.45	**37.20**	1.47
MDG-a_38_n2000_m200	2000	42.50	55.70	50.90	53.83	39.50	**36.60**	1.11
MDG-a_39_n2000_m200	2000	42.35	53.70	50.55	53.48	39.45	**36.85**	1.31
MDG-a_40_n2000_m200	2000	42.15	55.25	50.45	54.03	39.45	**37.45**	1.20
Avg	2000	42.56	54.51	50.78	53.66	39.64	**37.16**	1.27
#Better		0	0	0	0	0	**20**	
p-value		7.74e-06	7.74e-06	7.74e-06	7.74e-06	7.74e-06		

Table 6
Computational results and comparison in the best objective value obtained (f_{best}) on the MDG-b instances with $n = 2000$.

Instance	Time (s)	Best known	CIH [3]	EPR [13]	VNS [22]	ILS [33]	TS [27]	IDTS (this work)	
			f_{best}	f_{best}	f_{best}	f_{best}	f_{best}	f_{best}	Δf_{best}
MDG-b_21_n2000_m200	2000	3421.21	3592.78	4600.85	4232.27	3978.52	3421.21	**2980.75***	-440.46
MDG-b_22_n2000_m200	2000	3389.63	3610.15	4333.36	4280.79	3911.34	3420.91	**2961.21***	-428.42
MDG-b_23_n2000_m200	2000	3445.18	3608.12	4566.91	4196.89	4127.34	3448.59	**3074.56***	-370.62
MDG-b_24_n2000_m200	2000	3305.12	3599.84	4483.36	4188.47	4088.26	3305.12	**3007.62***	-297.50
MDG-b_25_n2000_m200	2000	3360.30	3527.50	4429.91	4362.02	3892.67	3360.30	**3062.53***	-297.77
MDG-b_26_n2000_m200	2000	3342.92	3644.37	4523.01	4145.28	4116.90	3534.09	**3068.00***	-274.92
MDG-b_27_n2000_m200	2000	3361.44	3693.03	4533.26	4068.17	4126.90	3361.44	**3103.56***	-257.88
MDG-b_28_n2000_m200	2000	3454.52	3643.33	4389.26	4195.74	4112.43	3454.52	**3091.04***	-363.48
MDG-b_29_n2000_m200	2000	3351.36	3707.34	4400.64	4039.83	4057.62	3457.26	**3046.27***	-305.09
MDG-b_30_n2000_m200	2000	3373.50	3678.40	4349.86	4270.79	4110.61	3373.50	**3041.00***	-332.50
MDG-b_31_n2000_m200	2000	3519.23	3752.73	4313.65	4083.42	4074.80	3519.23	**3040.03***	-479.20
MDG-b_32_n2000_m200	2000	3442.42	3673.65	4315.46	4240.51	3929.49	3442.42	**3060.99***	-381.43
MDG-b_33_n2000_m200	2000	3444.89	3706.50	4385.88	4387.52	3985.32	3444.89	**3061.50***	-383.39
MDG-b_34_n2000_m200	2000	3454.03	3773.05	4632.31	4113.29	4084.46	3454.03	**3071.88***	-382.15
MDG-b_35_n2000_m200	2000	3372.26	3699.91	4429.15	4119.50	4000.31	3457.00	**3055.21***	-317.05
MDG-b_36_n2000_m200	2000	3442.17	3715.52	4321.26	4131.32	4095.13	3442.17	**3050.39***	-391.78
MDG-b_37_n2000_m200	2000	3352.08	3664.97	4549.56	4232.38	4035.74	3458.43	**3015.38***	-336.70
MDG-b_38_n2000_m200	2000	3390.50	3661.20	4476.97	4295.61	4126.69	3390.50	**3104.92***	-285.58
MDG-b_39_n2000_m200	2000	3476.10	3672.97	4470.91	4114.55	4131.87	3476.10	**2900.08***	-576.02
MDG-b_40_n2000_m200	2000	3351.17	3719.84	4426.71	4136.50	4306.02	3375.62	**3016.38***	-334.79
Avg.		3402.50	3667.26	4446.61	4191.74	4064.62	3429.87	**3040.67**	-361.84
#Best		0	0	0	0	0	0	**20**	
p-value		7.74e-6	7.74e-6	7.74e-6	7.74e-6	7.74e-6	7.74e-6		

Table 7
Computational results and comparison in the average objective value obtained (f_{avg}) on the MDG-b instances with $n = 2000$.

		CIH [3]	EPR [13]	VNS [22]	ILS [33]	TS [27]	IDTS (this work)	
Instance	Time (s)	f_{avg}	f_{avg}	f_{avg}	f_{avg}	f_{avg}	f_{avg}	*std.*
MDG-b_21_n2000_m200	2000	3883.27	4778.31	4435.83	4299.38	3544.32	**3280.31**	114.60
MDG-b_22_n2000_m200	2000	3879.67	4661.84	4520.33	4377.97	3564.41	**3274.61**	91.25
MDG-b_23_n2000_m200	2000	3808.08	4722.15	4390.30	4422.12	3550.02	**3295.18**	102.89
MDG-b_24_n2000_m200	2000	3839.34	4707.11	4472.02	4421.77	3532.08	**3282.48**	112.17
MDG-b_25_n2000_m200	2000	3825.67	4794.93	4557.13	4340.78	3603.87	**3268.85**	85.49
MDG-b_26_n2000_m200	2000	3880.27	4730.99	4391.32	4423.07	3630.28	**3292.18**	104.27
MDG-b_27_n2000_m200	2000	3868.30	4701.02	4385.32	4424.59	3530.74	**3305.33**	91.60
MDG-b_28_n2000_m200	2000	3810.18	4698.69	4477.90	4446.16	3545.25	**3275.35**	104.37
MDG-b_29_n2000_m200	2000	3870.87	4681.13	4301.16	4377.08	3553.72	**3289.42**	108.10
MDG-b_30_n2000_m200	2000	3797.06	4764.17	4420.86	4470.64	3547.15	**3288.46**	92.69
MDG-b_31_n2000_m200	2000	3861.12	4801.32	4415.22	4323.11	3609.88	**3272.11**	102.03
MDG-b_32_n2000_m200	2000	3797.78	4778.58	4366.35	4301.35	3566.98	**3276.19**	101.40
MDG-b_33_n2000_m200	2000	3815.30	4697.26	4574.32	4351.01	3584.87	**3271.92**	109.81
MDG-b_34_n2000_m200	2000	3894.40	4791.64	4529.20	4402.11	3578.48	**3292.90**	110.45
MDG-b_35_n2000_m200	2000	3883.25	4728.08	4342.11	4396.43	3580.56	**3290.86**	115.81
MDG-b_36_n2000_m200	2000	3897.08	4653.35	4356.16	4435.33	3574.16	**3247.02**	103.31
MDG-b_37_n2000_m200	2000	3857.85	4836.76	4381.58	4409.06	3593.93	**3331.37**	108.89
MDG-b_38_n2000_m200	2000	3803.77	4685.33	4405.56	4418.53	3572.96	**3278.91**	112.47
MDG-b_39_n2000_m200	2000	3863.94	4698.42	4291.46	4403.46	3590.59	**3274.59**	123.41
MDG-b_40_n2000_m200	2000	3816.35	4670.78	4391.52	4306.02	3523.60	**3281.05**	124.97
Avg.		3847.68	4729.09	4420.28	4387.50	3568.89	**3283.46**	106.00
#Best		0	0	0	0	0	**20**	
p-value		7.74e-6	7.74e-6	7.74e-6	7.74e-6	7.74e-6		

Table 8
Computational results and comparison in the best objective value obtained (f_{best}) on the MDG-c instances with $n = 3000$.

			CIH [3]	EPR [13]	VNS [22]	ILS [33]	TS [27]	IDTS (this work)	
Instance	Time (s)	Best known	f_{best}	f_{best}	f_{best}	f_{best}	f_{best}	f_{best}	Δf_{best}
MDG-c_1_n3000_m300	3000	4796	5215	6661	6145	5772	4796	**4583***	-213
MDG-c_2_n3000_m300	3000	4827	5203	6482	5975	5936	4830	**4542***	-285
MDG-c_3_n3000_m300	3000	4913	5174	6518	6105	5585	4913	**4317***	-596
MDG-c_4_n3000_m300	3000	4830	5164	6245	6465	5969	4830	**4385***	-445
MDG-c_5_n3000_m300	3000	4809	5175	6500	6152	5750	4881	**4641***	-168
MDG-c_6_n3000_m400	3000	6349	6883	8646	8313	7648	6466	**6028***	-321
MDG-c_7_n3000_m400	3000	6334	6916	8016	7890	7829	6480	**5725***	-609
MDG-c_8_n3000_m400	3000	6255	7417	8198	8248	7984	6255	**5993***	-262
MDG-c_9_n3000_m400	3000	6346	6652	8321	8298	7657	6607	**5863***	-483
MDG-c_10_n3000_m400	3000	6297	6797	9206	8514	7672	6297	**5959***	-338
MDG-c_11_n3000_m500	3000	7793	8477	10130	10236	11031	7793	**7539***	-254
MDG-c_12_n3000_m500	3000	7719	8293	10081	10428	10604	7719	**7538***	-181
MDG-c_13_n3000_m500	3000	7711	8078	10847	10318	10743	7767	**7480***	-231
MDG-c_14_n3000_m500	3000	**7645**	8470	10472	10327	9941	7678	7739	94
MDG-c_15_n3000_m500	3000	7659	8536	10489	10320	10870	7659	**7511***	-148
MDG-c_16_n3000_m600	3000	9337	10066	12104	12007	13910	9337	**8680***	-657
MDG-c_17_n3000_m600	3000	**8618**	10091	13924	12083	13676	**8618**	8997	379
MDG-c_18_n3000_m600	3000	9118	10451	13322	12538	14011	9118	**8978***	-140
MDG-c_19_n3000_m600	3000	9387	12313	12329	12216	13538	9387	**8686***	-701
MDG-c_20_n3000_m600	3000	**9013**	10284	12219	12231	12415	**9013**	9079	66
Avg	3000	6987.80	7782.75	9535.50	9240.45	9427.05	7022.20	**6713.15**	-274.65
#Best		3	0	0	0	0	3	17	
p-value		1.75e-03	7.74e-06	7.74e-06	7.74e-06	7.74e-06	1.75e-03		

Table 9
Computational results and comparison in the average objective value obtained (f_{avg}) on the MDG-c instances with $n = 3000$.

		CIH [3]	EPR [13]	VNS [22]	ILS [33]	TS [27]	IDTS (this work)	
Instance	Time (s)	f_{avg}	f_{avg}	f_{avg}	f_{avg}	f_{avg}	f_{avg}	*std.*
MDG-c_1_n3000_m300	3000	5537.60	7139.85	6393.85	6265.60	5018.60	**4772.90**	103.49
MDG-c_2_n3000_m300	3000	5393.10	7197.70	6378.40	6539.33	5020.70	**4772.60**	128.96
MDG-c_3_n3000_m300	3000	5604.60	7294.30	6545.25	6243.03	5107.45	**4740.50**	215.89
MDG-c_4_n3000_m300	3000	5493.75	7152.85	6723.30	6636.75	4988.05	**4689.20**	199.65
MDG-c_5_n3000_m300	3000	5431.60	6845.75	6290.95	6663.25	5118.75	**4832.70**	142.36
MDG-c_6_n3000_m400	3000	7599.85	9513.10	8714.50	8412.98	6680.65	**6351.20**	171.66
MDG-c_7_n3000_m400	3000	7763.75	9273.25	8690.90	8457.15	6855.30	**6382.45**	259.46
MDG-c_8_n3000_m400	3000	7894.35	9258.80	8566.05	8497.28	6518.55	**6294.00**	167.27
MDG-c_9_n3000_m400	3000	7027.35	9116.20	8651.60	8259.35	6913.70	**6341.30**	226.54
MDG-c_10_n3000_m400	3000	7188.35	10022.30	8912.15	8646.00	6469.70	**6266.40**	225.11
MDG-c_11_n3000_m500	3000	9086.55	11486.05	10896.90	12223.38	8064.00	**7877.45**	201.53
MDG-c_12_n3000_m500	3000	8927.50	11965.35	10735.35	12103.03	8101.60	**7905.85**	242.39
MDG-c_13_n3000_m500	3000	9207.35	12232.10	10692.20	12228.58	8206.10	**7993.10**	299.98
MDG-c_14_n3000_m500	3000	8859.75	12394.55	10885.55	11643.90	8114.90	**7946.15**	154.03
MDG-c_15_n3000_m500	3000	9174.90	11945.55	11032.65	12365.85	7991.05	**7895.05**	212.32
MDG-c_16_n3000_m600	3000	11516.70	13846.90	12406.05	15801.65	9878.05	**9505.65**	352.73
MDG-c_17_n3000_m600	3000	11226.35	14663.65	12978.90	15284.10	**9529.30**	9601.40	285.28
MDG-c_18_n3000_m600	3000	11098.75	14411.05	13077.40	15547.08	9540.30	**9502.25**	305.41
MDG-c_19_n3000_m600	3000	13038.15	14364.90	12870.45	15526.85	9696.40	**9360.80**	367.25
MDG-c_20_n3000_m600	3000	11390.65	13966.90	12707.40	13545.33	9618.75	**9550.30**	265.14
Avg.		8423.05	10704.56	9707.49	10544.52	7371.60	**7129.06**	226.32
#Best		0	0	0	0	1	19	
p-value		7.74e-06	7.74e-06	7.74e-06	7.74e-06	5.70e-05		

algorithm dominates the reference algorithms in terms of f_{avg}, where the IDTS algorithm obtained a better result for all 20 instances. The associated standard deviations (std) are very small for all instances (≤ 2.0). The superiority of the IDTS algorithm over the reference algorithms is also confirmed by the small $p - values$ (≤ 0.05) both in terms of f_{best} and f_{avg}.

Tables 4 and 5 show that for the MDG-a instances with $n = 2000$ our IDTS algorithm significantly outperforms the five state-of-the-art algorithms both in terms of f_{best} and f_{avg}. Specifically, the IDTS algorithm improved the best known results in the literature for all 20 instances and also obtained better f_{avg} values on all instances. The significance of the differences between the results of the IDTS algorithm and those of the reference algorithms is again confirmed by the small $p - values$ (< 0.05). Furthermore, the standard deviations (std) are less than 2.0, implying a good robustness of the IDTS algorithm.

Tables 6 and 7 show that for the large-scale MDG-b instances with $n = 2000$ our IDTS algorithm improved the previous best known results for all 20 instances, and obtained better results both in terms of f_{best} and f_{avg} for all 20 instances compared to any of the five reference algorithms.

Tables 8 and 9 show the computational results of our IDTS algorithm and the five reference algorithms on the MDG-c instances. Table 8 shows that the IDTS algorithm improved the previous best known result in the literature for 17 out of 20 instances, and missed the previous best known results for only 3 instances. Compared to the latest TS algorithm of [27], the IDTS algorithm yielded a better and worse result in terms of f_{avg} for 17 and 3 instances, respectively. Compared to the other 4 reference algorithms, IDTS yielded a better result for all 20 instances. Table 9 indicates that IDTS outperforms the TS algorithm of [27] for 19 out of 20 instances in terms of f_{avg}, and outperforms the other four reference algorithms for all 20 instances. Once again, the significance of the differences between the results of the IDTS algorithm and those of the reference algorithms is confirmed by *p-values* less than 0.05.

In summary, the above comparative studies disclose that our IDTS algorithm compares very favorably with the state-of-the-art Min-Diff DP algorithms in the literature.

4 Analysis and Discussions

We analyse and discuss several essential features of the IDTS algorithm to understand their impacts on the performance, including the sensitivity of the key parameters, the effectiveness of the intensified search mechanism and the constrained neighborhood. In addition, based on some representative instances, we analyse the moving trajectory of the IDTS algorithm and the spacial distribution of high-quality solutions to shed light on the landscape of Min-Diff DP.

4.1 Analysis of the Key Parameters

Table 10
Influence of the parameter α on the performance of the IDTS algorithm. The best *Avg* result is indicated in bold.

	P1	P2	P3	P4	
α	f_{avg}	f_{avg}	f_{avg}	f_{avg}	Avg
5	1253.80	3490.00	3533.54	5085.20	3340.63
10	1150.48	3372.28	3309.15	4686.80	3129.68
15	1127.10	3248.64	3317.53	4669.95	3090.80
20	1127.75	3250.34	3254.51	4680.85	3078.36
25	1109.77	3296.11	3295.88	4653.65	3088.85
30	1112.58	3290.97	3252.77	4821.05	3119.34
35	1131.17	3270.20	3288.31	4620.25	**3077.48**
40	1110.93	3366.32	3315.90	4769.90	3140.76
45	1106.34	3258.68	3297.83	4740.45	3100.82
50	1094.36	3284.21	3307.38	4808.65	3123.65
60	1110.30	3324.87	3347.71	4819.50	3150.60
100	1093.88	3359.40	3351.72	4695.05	3125.01

As previously indicated, the IDTS algorithm employs two key parameters, the value α that fixes the maximum number of non-improving tabu search iterations with respect to the recorded best solution s^* and the value θ that controls

Table 11
Influence of the parameter θ on the performance of the IDTS algorithm. The best *Avg* result is indicated in bold.

	P1	P2	P3	P4	
θ	f_{avg}	f_{avg}	f_{avg}	f_{avg}	Avg
0.05	1259.94	3488.03	3490.39	4892.30	3282.67
0.10	1189.86	3417.34	3403.95	4815.10	3206.56
0.15	1162.95	3374.28	3350.06	4725.45	3153.19
0.20	1116.08	3289.13	3357.32	4740.90	3125.86
0.25	1119.22	3323.78	3334.07	4743.35	3130.11
0.30	1110.81	3320.30	3332.74	4703.85	**3116.93**
0.35	1110.53	3332.74	3331.70	4765.85	3135.21
0.40	1110.93	3366.32	3315.90	4769.90	3140.76
0.45	1116.06	3382.50	3319.98	4781.30	3149.96
0.50	1100.71	3391.71	3342.26	4877.05	3177.93
0.55	1134.28	3341.03	3390.44	4901.95	3191.92
0.60	1104.73	3331.52	3340.25	4870.10	3161.65

the size of neighborhood N^{θ}_{swap}. To investigate the influence of α, we carried out an experiment on 4 representative instances MDG-b_1_n500_m50, MDG-b_21_n2000_m200, MDG-b_40_n2000_m200, and MDG-c_1_n3000_m300 that are renamed as 'P1', 'P2', 'P3', and 'P4' for simplicity. For each α value in $\{5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 100\}$, we solved each instance 20 times, using the experimental protocol in Section 3.2. The computational results are summarized in Table 10, where the first column shows the setting of α, the last column shows the average results over all instances (*Avg*), and other columns give the average objective values over 20 runs for each instance. Table 10 shows that no α value performs the best on all instances and that a medium α value leads generally to a globally acceptable performance, while large and small α values lead to a large performance difference on different instances. Hence, as a comprise, we adopt $\alpha = 35$ as the default value for our IDTS algorithm.

To check whether the performance of the algorithm is sensitive to the setting of θ, we carried out another experiment based on the 4 representative instances mentioned above. For each instance and each θ value in {0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6}, the IDTS algorithm was run 20 times, and the computational results are summarized in Table 11. We observe from Table 11 that similar to the parameter α, a medium θ value leads to an acceptable performance of the algorithm on all instances tested. The last column of the table shows that the setting $\theta = 0.3$ produced the best outcome in terms of *Avg* among all tested settings. As a result, the default value of θ is set to 0.3 for our IDTS algorithm.

4.2 Effectiveness of the Constrained Neighborhood

The constrained swap neighborhood N^{θ}_{swap} used as a candidate list strategy is an essential component of the IDTS algorithm. To study the effectiveness

Table 12
Comparative results of the constrained swap neighborhood N^{θ}_{swap} with the full swap neighborhood N^{full}_{swap} on the 20 large instances of set MDG-b.

Instance	Time (s)	f_{best}		f_{avg}		f_{worst}	
		IDTS*	IDTS	IDTS*	IDTS	IDTS*	IDTS
MDG-b_21_n2000_m200	2000	3227.73	**2980.75**	3554.41	**3280.31**	3774.53	**3497.11**
MDG-b_22_n2000_m200	2000	3203.54	**2961.21**	3424.94	**3274.61**	3632.66	**3455.44**
MDG-b_23_n2000_m200	2000	3281.86	**3074.56**	3495.20	**3295.18**	3779.58	**3588.02**
MDG-b_24_n2000_m200	2000	3181.18	**3007.62**	3517.09	**3282.48**	3707.87	**3557.54**
MDG-b_25_n2000_m200	2000	3326.85	**3062.53**	3525.38	**3268.85**	3764.58	**3453.27**
MDG-b_26_n2000_m200	2000	3298.21	**3068.00**	3532.70	**3292.18**	3746.72	**3506.27**
MDG-b_27_n2000_m200	2000	3267.52	**3103.56**	3524.25	**3305.33**	3843.87	**3479.93**
MDG-b_28_n2000_m200	2000	3331.40	**3091.04**	3520.57	**3275.35**	3827.14	**3541.91**
MDG-b_29_n2000_m200	2000	3137.31	**3046.27**	3498.12	**3289.42**	3766.85	**3656.07**
MDG-b_30_n2000_m200	2000	3248.86	**3041.00**	3535.45	**3288.46**	3793.35	**3469.45**
MDG-b_31_n2000_m200	2000	3301.59	**3040.03**	3522.19	**3272.11**	3822.31	**3506.72**
MDG-b_32_n2000_m200	2000	3179.60	**3060.99**	3515.59	**3276.19**	3756.51	**3495.65**
MDG-b_33_n2000_m200	2000	3205.76	**3061.50**	3491.72	**3271.92**	3734.97	**3525.80**
MDG-b_34_n2000_m200	2000	3100.92	**3071.88**	3496.86	**3292.90**	3788.15	**3487.91**
MDG-b_35_n2000_m200	2000	3385.95	**3055.21**	3555.96	**3290.86**	3763.23	**3601.60**
MDG-b_36_n2000_m200	2000	3314.21	**3050.39**	3545.67	**3247.02**	3807.67	**3450.08**
MDG-b_37_n2000_m200	2000	3227.34	**3015.38**	3478.66	**3331.37**	3691.13	**3512.72**
MDG-b_38_n2000_m200	2000	3272.18	**3104.92**	3535.02	**3278.91**	3781.62	**3528.55**
MDG-b_39_n2000_m200	2000	3275.65	**2900.08**	3529.54	**3274.59**	3820.13	**3510.92**
MDG-b_40_n2000_m200	2000	3206.93	**3016.38**	3452.30	**3281.05**	3652.17	**3597.83**
#Better		0	20	0	20	0	20
#Equal		0	0	0	0	0	0
#Worse		20	0	20	0	20	0
p-value			7.74e-06		7.74e-06		7.74e-06

of this strategy, we created a variant of the IDTS algorithm called IDTS* by replacing the constrained swap neighborhood N^{θ}_{swap} by the full swap neighborhood N^{full}_{swap}, while keeping other components of the IDTS algorithm unchanged. Then, we carried out an experiment based on the 20 large MDG-b instances with $n = 2000$ and $m = 200$, executing the IDTS* and IDTS algorithms 20 times on each instance according to the experimental protocol of Section 3.2.

The computational results of this experiment are summarized in Table 12, including the time limits used, the best (f_{best}), average (f_{avg}) and worst (f_{worst}) objective values. The rows $\#Better$, $\#Equal$ and $\#Worse$ show the numbers of instances for which each algorithm yielded a better result than the other algorithm in terms of f_{best}, f_{avg}, and f_{worst}. To verify whether there exists a significant difference between the results obtained by the compared algorithms, the *p-values* from the non-parametric Friedman tests are provided in the last row.

Table 12 shows that IDTS (with the constrained neighborhood N^{θ}_{swap}) consistently outperforms IDTS* (with the full neighborhood N^{full}_{swap}) on all 20 instances in terms of f_{best}, f_{avg}, and f_{worst}, confirming that the constrained swap neighborhood N^{θ}_{swap} plays a positive role in enhancing algorithmic performance on the tested instances given the time limits employed. On the other hand, the effectiveness of N^{θ}_{swap} also depends on the setting of the parameter θ, as demonstrated in Section 4.1.

4.3 *Effectiveness of the Intensified Search Mechanism*

Table 13
Comparative results of the IDTS algorithm with and without the intensified search mechanism on the 20 large instances of set MDG-b.

		f_{best}		f_{avg}		f_{worst}	
Instance	Time (s)	IDTS$^-$	IDTS	IDTS$^-$	IDTS	IDTS$^-$	IDTS
MDG-b_21_n2000_m200	2000	3531.82	**2980.75**	3607.87	**3280.31**	3689.28	**3497.11**
MDG-b_22_n2000_m200	2000	3425.31	**2961.21**	3581.12	**3274.61**	3702.27	**3455.44**
MDG-b_23_n2000_m200	2000	3435.43	**3074.56**	3589.84	**3295.18**	3692.52	**3588.02**
MDG-b_24_n2000_m200	2000	3296.40	**3007.62**	3593.57	**3282.48**	3709.41	**3557.54**
MDG-b_25_n2000_m200	2000	3474.71	**3062.53**	3645.80	**3268.85**	3725.34	**3453.27**
MDG-b_26_n2000_m200	2000	3476.76	**3068.00**	3597.27	**3292.18**	3718.05	**3506.27**
MDG-b_27_n2000_m200	2000	3430.97	**3103.56**	3592.84	**3305.33**	3706.95	**3479.93**
MDG-b_28_n2000_m200	2000	3513.96	**3091.04**	3622.38	**3275.35**	3727.75	**3541.91**
MDG-b_29_n2000_m200	2000	3536.59	**3046.27**	3607.95	**3289.42**	3701.91	**3656.07**
MDG-b_30_n2000_m200	2000	3461.98	**3041.00**	3602.71	**3288.46**	3740.34	**3469.45**
MDG-b_31_n2000_m200	2000	3493.03	**3040.03**	3578.02	**3272.11**	3665.83	**3506.72**
MDG-b_32_n2000_m200	2000	3401.52	**3060.99**	3593.41	**3276.19**	3715.99	**3495.65**
MDG-b_33_n2000_m200	2000	3455.67	**3061.50**	3622.39	**3271.92**	3758.12	**3525.80**
MDG-b_34_n2000_m200	2000	3378.85	**3071.88**	3560.27	**3292.90**	3732.65	**3487.91**
MDG-b_35_n2000_m200	2000	3516.59	**3055.21**	3636.91	**3290.86**	3735.21	**3601.60**
MDG-b_36_n2000_m200	2000	3504.46	**3050.39**	3626.13	**3247.02**	3762.41	**3450.08**
MDG-b_37_n2000_m200	2000	3403.84	**3015.38**	3587.46	**3331.37**	3708.17	**3512.72**
MDG-b_38_n2000_m200	2000	3336.39	**3104.92**	3586.67	**3278.91**	3745.11	**3528.55**
MDG-b_39_n2000_m200	2000	3458.21	**2900.08**	3617.42	**3274.59**	3747.81	**3510.92**
MDG-b_40_n2000_m200	2000	3449.57	**3016.38**	3620.62	**3281.05**	3714.19	**3597.83**
#Better		0	20	0	20	0	20
#Equal		0	0	0	0	0	0
#Worse		20	0	20	0	20	0
p-value			7.74e-06		7.74e-06		7.74e-06

The intensified search mechanism is another essential component of the proposed IDTS algorithm for the purpose of intensifying the search around the last best solution found. To study its impacts on the performance of IDTS, we created a variant of the IDTS algorithm called IDTS$^-$, where we disabled the intensified search mechanism (line 7 of Algorithm 1), while keeping other components unchanged. As in Section 4.2, we compare IDTS and IDTS$^-$ based on the 20 large instances with $n = 2000$ and $m = 200$ of the set MDG-b. We ran both IDTS$^-$ and IDTS 20 times to solve each instance, using the experimental protocol of Section 3.2.

The experimental results are summarized in Table 13, where we include the same statistics as in Table 12. Table 13 clearly shows that the IDTS algorithm (with the intensified search mechanism) performs consistently much better than IDTS$^-$ (without the intensified search mechanism) over all performance indicators considered and on all the tested instances, as confirmed by the small *p-values*. This outcome demonstrates that the intensified search mechanism plays a highly positive role in the high performance of the IDTS algorithm.

4.4 *Influence of Hash Vectors and Hash Functions*

The proposed IDTS algorithm uses three hash vectors of length $L = 10^8$ to manage the tabu list (see Section 2.5). To investigate the influence of these

Table 14
Experimental results of the proposed algorithm with different numbers of hash vectors and different lengths (L) of hash vectors, where the average objective value (f_{avg}) over 20 runs is reported for each instance and each setting.

Instance	Two Hash Vectors ($L = 10^8$)			Three Hash Vectors		
	IDTS$_1$ (H_1, H_2)	IDTS$_2$ (H_1, H_3)	IDTS$_3$ (H_2, H_3)	IDTS$_4$ ($L = 10^6$)	IDTS$_5$ ($L = 10^7$)	IDTS ($L = 10^8$)
MDG-b_1_n500_m50	1095.38	1090.80	1113.68	1128.85	1092.90	1109.54
MDG-b_2_n500_m50	1111.31	1101.85	1105.09	1094.83	1109.63	1101.90
MDG-b_3_n500_m50	1135.32	1104.65	1124.82	1099.51	1105.00	1113.33
MDG-b_4_n500_m50	1117.34	1115.78	1107.98	1132.73	1101.97	1106.83
MDG-b_5_n500_m50	1112.37	1102.89	1112.15	1114.05	1102.48	1110.93
MDG-b_6_n500_m50	1126.47	1113.69	1122.27	1123.82	1118.33	1108.56
MDG-b_7_n500_m50	1109.36	1120.34	1114.56	1100.51	1106.37	1121.52
MDG-b_8_n500_m50	1115.28	1104.25	1120.91	1120.48	1118.54	1122.64
MDG-b_9_n500_m50	1122.09	1110.42	1122.27	1113.20	1113.18	1116.71
MDG-b_10_n500_m50	1106.08	1109.63	1123.60	1115.00	1116.72	1116.91
MDG-b_11_n500_m50	1129.84	1118.48	1116.27	1100.90	1106.86	1124.39
MDG-b_12_n500_m50	1113.66	1120.70	1108.58	1116.99	1115.64	1095.78
MDG-b_13_n500_m50	1135.50	1118.32	1115.74	1094.78	1120.83	1092.17
MDG-b_14_n500_m50	1118.15	1122.20	1117.64	1113.11	1123.09	1108.42
MDG-b_15_n500_m50	1109.67	1124.51	1104.98	1103.18	1106.04	1104.19
MDG-b_16_n500_m50	1111.01	1107.44	1094.62	1136.58	1123.35	1092.32
MDG-b_17_n500_m50	1102.21	1113.53	1120.63	1124.57	1101.54	1137.81
MDG-b_18_n500_m50	1105.21	1103.19	1126.20	1116.62	1108.77	1105.58
MDG-b_19_n500_m50	1121.57	1116.59	1104.55	1108.09	1110.67	1114.25
MDG-b_20_n500_m50	1123.84	1111.71	1101.14	1104.99	1106.75	1116.59
Avg.	1116.08	1111.55	1113.88	1113.14	1110.43	1111.02

elements, we first created three variants IDTS$_1$, IDTS$_2$ and IDTS$_3$ by disabling the hash vectors H_3, H_2, and H_1 of IDTS, respectively, while keeping other components of algorithm unchanged. We also created two other variants IDTS$_4$ and IDTS$_5$ of the IDTS algorithm where we replace the default length of hash vectors ($L = 10^8$) by $L = 10^6$ and $L = 10^7$ respectively. Then, we carried out an experiment on the 20 MDG-b instances with $n = 500$ by running each of these variants 20 times to solve each instance according to the experimental protocol in Section 3.2.

Columns 2–4 of Table 14 show that under the current experimental conditions, IDTS performs similarly with two or three hash vectors in terms of the average results for the tested instances. Nevertheless, given that 1) using more hash vectors theoretically helps to reduce the number of possible collisions in the general case, and 2) determining the tabu status of a neighbor solution has a very low time complexity (bounded by $O(1)$) when using either two or three hash vectors, we adopt three hash vectors in our IDTS algorithm. A similar observation can be made for IDTS$_4$ and IDTS$_5$, which indicates that IDTS is not sensitive to the length (L) of hash vectors.

As shown in Section 2.5, the hash functions involve a parameter (ξ_k, $k = 1, 2, 3$), each parameter ξ_k leading to a hash function h_k. To show the influence of hash functions on the performance of the IDTS algorithm, we carried

Table 15. Experimental results of IDTS with 9 parameter combinations of (ξ_1, ξ_2, ξ_3) (hash functions), in terms of the average objective values (f_{avg}) over 20 runs. The best results among those obtained by the tested parameter combinations are indicated in bold for each instance.

	f_{avg}								
Instance/(ξ_1, ξ_2, ξ_3)	(1.1, 1.2, 1.3)	(1.1, 1.2, 1.5)	(1.1, 1.3, 1.5)	(1.1, 1.3, 1.9)	(1.1, 1.4, 2.0)	(1.1, 1.5, 2.0)	(1.5, 1.8, 1.9)	(1.8, 1.9, 2.0)	(2.0, 2.1, 2.2)
MDG-b_1_n500_m50	1197.51	1175.63	1168.79	1123.11	1132.04	1143.28	**1096.23**	1109.54	1106.88
MDG-b_2_n500_m50	1204.43	1169.34	1157.87	1129.67	1131.82	1129.48	1117.85	**1101.90**	1107.47
MDG-b_3_n500_m50	1204.65	1170.33	1161.84	1117.45	1127.47	1127.48	1122.84	**1113.33**	1124.10
MDG-b_4_n500_m50	1203.04	1154.75	1168.29	**1102.95**	1113.68	1123.58	1106.84	1106.83	1115.82
MDG-b_5_n500_m50	1216.52	1155.96	1154.67	1130.46	1117.23	1103.90	**1100.49**	1110.93	1107.77
MDG-b_6_n500_m50	1205.84	1176.52	1155.93	1122.89	1125.39	1110.60	1116.93	**1108.56**	1117.50
MDG-b_7_n500_m50	1201.84	1163.48	1159.13	1123.49	1122.18	1108.56	1113.28	1121.52	**1107.91**
MDG-b_8_n500_m50	1202.44	1180.83	1160.94	**1109.50**	1121.61	1130.28	1124.86	1122.64	1115.86
MDG-b_9_n500_m50	1182.80	1171.06	1185.53	1126.07	1120.59	1114.03	1123.41	1116.71	**1113.71**
MDG-b_10_n500_m50	1196.40	1166.65	1162.30	1124.79	1118.92	**1109.98**	1126.91	1116.91	1135.17
MDG-b_11_n500_m50	1212.28	1187.99	1147.17	1126.70	1118.42	1104.80	1111.05	1124.39	**1103.96**
MDG-b_12_n500_m50	1199.01	1153.81	1172.42	1123.18	1110.24	1122.80	1101.50	**1095.78**	1121.16
MDG-b_13_n500_m50	1184.25	1175.17	1146.24	1115.59	1116.61	1120.68	1094.41	1092.17	**1085.32**
MDG-b_14_n500_m50	1208.96	1159.32	1170.90	**1096.59**	1137.25	1136.10	1099.84	1108.42	1133.12
MDG-b_15_n500_m50	1178.70	1172.74	1150.44	1126.27	1111.18	1129.91	1121.54	1104.19	**1102.03**
MDG-b_16_n500_m50	1199.96	1168.81	1168.87	1123.22	1103.65	1138.99	1108.76	**1092.32**	1102.56
MDG-b_17_n500_m50	1186.48	1161.27	1173.35	1137.26	1116.24	1124.42	**1098.84**	1137.81	1116.57
MDG-b_18_n500_m50	1192.59	1188.32	1142.87	1131.88	1113.69	1120.07	**1102.32**	1105.58	1131.50
MDG-b_19_n500_m50	1189.19	1180.93	1156.06	**1109.78**	1121.52	1124.68	1120.75	1114.25	1119.38
MDG-b_20_n500_m50	1186.61	1179.53	1171.94	1120.93	1124.43	1111.96	1112.28	1116.59	**1107.60**
Avg.	1197.68	1170.62	1161.78	1121.09	1120.21	1121.78	1111.05	**1111.02**	1113.77
#Best	0	0	0	4	0	1	4	5	6

out an additional experiment to study the ξ_k parameter. For this purpose, we selected 9 representative parameter combinations (ξ_1, ξ_2, ξ_3) and ran the IDTS algorithm 20 times with each parameter combination to solve each of the 20 MDG-b instances. The average objective results (f_{avg}) are reported in Table 15, where the row *Avg.* shows the average result for each column and "#Best" shows the number of instances for which the corresponding parameter combination leads to the best result in terms of f_{avg}.

The results of Table 15 show that the performance of the IDTS algorithm is sensitive to the setting of parameters ξ_1, ξ_2 and ξ_3. For the parameter combinations containing a small value for all parameters, such as $(\xi_1, \xi_2, \xi_3) = (1.1, 1.2, 1.3), (1.1, 1.2, 1.5), (1.1, 1.3, 1.5)$, IDTS performs badly, yielding a worse result in terms of both "Avg." and "#Best" in comparison with other combinations. On the contrary, for those parameter combinations containing a large value for at least two parameters, such as $(1.5, 1.8, 1.9)$, $(1.8, 1.9, 2.0)$ and $(2.0, 2.1, 2.2)$, IDTS performs very well. As a result, for the present IDTS algorithm, the default combination of (ξ_1, ξ_2, ξ_3) is set to $(1.8, 1.9, 2.0)$, since such a setting led to the best result in terms of *Avg.* among the tested combinations.

4.5 Spatial Distribution of High-Quality Solutions

In an attempt to further understand why the intensified search mechanism is helpful, we have conducted a study on the spatial distribution of high-quality solutions as in [18,23]. Our experiment was based on 8 representative instances with $n = 2000$ or 3000, performing 10 runs of our IDTS algorithm for each instance tested, and then collecting all the high-quality local optimal solutions visited by the IDTS algorithm to characterize the spatial distribution of high-quality solutions. Here, a solution s is considered be of high-quality if its objective value $f(s)$ is better than $1.03 \times f_{bkv}$, i.e., $f(s) < 1.03 \times f_{bkv}$, where f_{bkv} represents the previous best known result in the literature. Following [18,23], to obtain a visual image of the spatial distribution of high-quality solutions obtained, we adopted the multidimensional scaling (MDS) method to generate approximately the distribution of solutions in the Euclidean space R^3 as follows. First, we generate a distance matrix $D_{l\times l}$, where l is the number of local optimum solutions sampled, and $d'_{ij} \in D_{l\times l}$ is the distance between solutions s_i and s_j. Specifically, given two solutions $s_i = (I_i^0, I_i^1)$ and $s_j = (I_j^0, I_j^1)$ of Min-Diff DP, the distance between s_i and s_j is calculated as $d'_{ij} = \frac{m-|I_i^1 \cap I_j^1|}{m}$. Then, according to the distance matrix obtained, we generate l coordinate points in the R^3 space by the *cmdscale* method, where the distance distortion between the obtained coordinate points is minimized. Finally, the scatter graph of the resulting points in R^3 is plotted. Interested readers are referred to [18,23] for more details of plotting the spatial distribution in the Euclidean

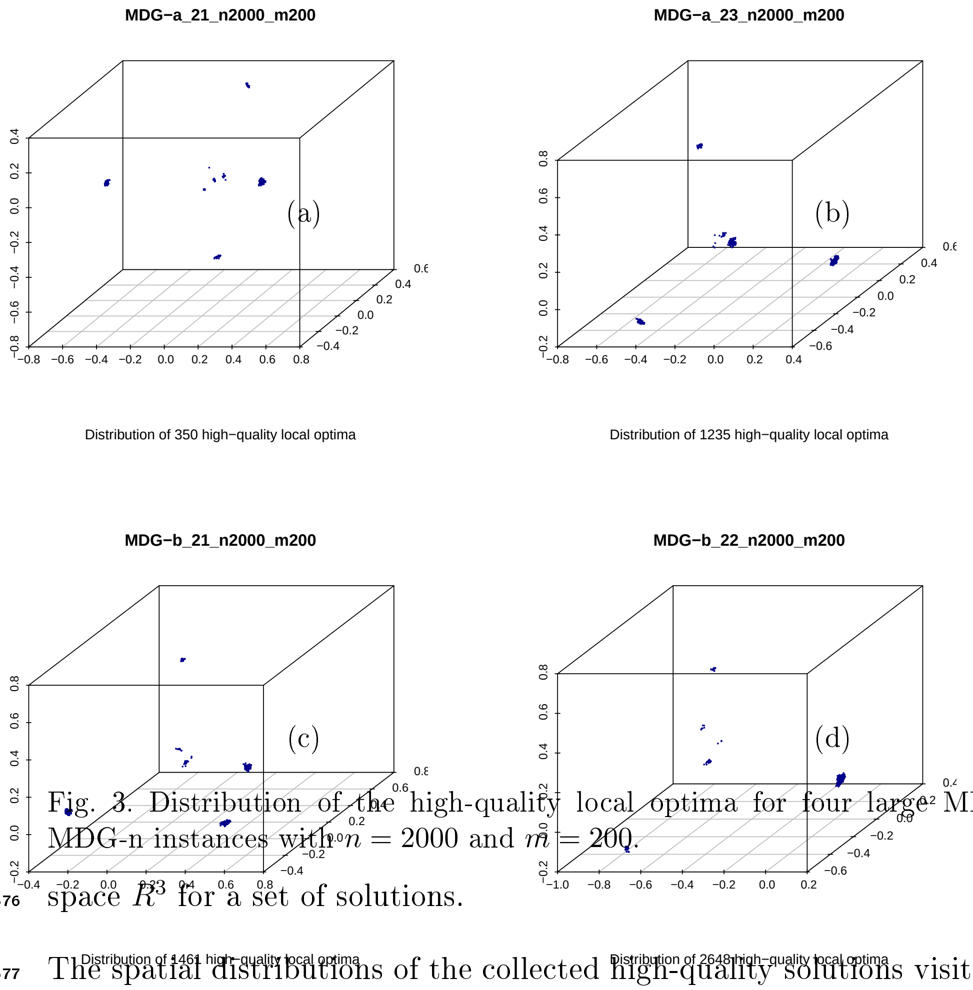

Fig. 3. Distribution of the high-quality local optima for four large MDG-a and MDG-n instances with $n = 2000$ and $m = 200$.

space R^3 for a set of solutions.

The spatial distributions of the collected high-quality solutions visited by the IDTS algorithm are given in Fig. 3 and Fig. 4 for the selected instances. First, these plots show that for all tested instances, the collected high-quality solutions are typically grouped in clusters, delimited by a sphere of small diameter and characterized by small distances between the solutions of the same cluster [23]. This observation implies that the solutions within a cluster can be reached more easily from a nearby solution than from a distant solution. The intensified search mechanism of the IDTS algorithm exploits this property by systematically launching a search from the best solution found so far in order to discover other nearby high-quality solutions. Second, to discover a new cluster (that can contain new high-quality solutions), it is useful to apply some strong diversification strategies. In the case of the IDTS algorithm, this is achieved by the simple mechanism of multiple re-starts, each re-start being

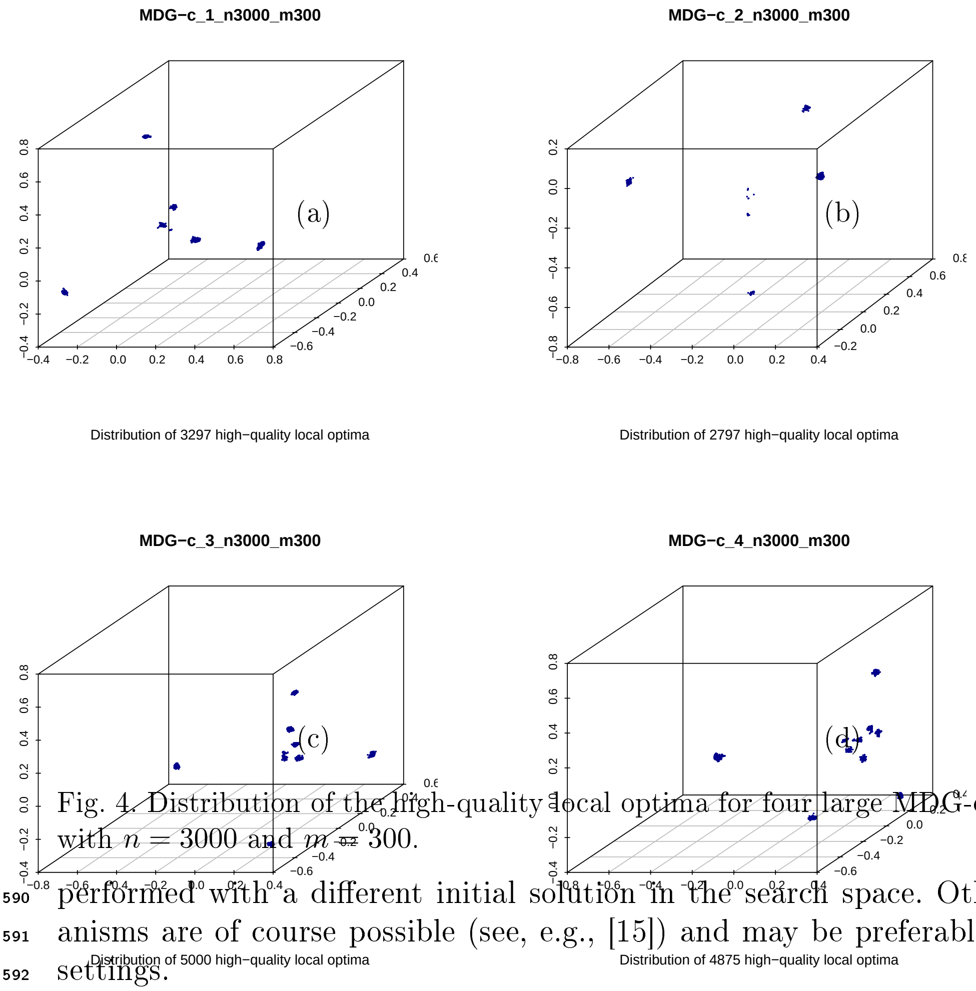

Fig. 4. Distribution of the high-quality local optima for four large MDG-c instances with $n = 3000$ and $m = 300$.

performed with a different initial solution in the search space. Other mechanisms are of course possible (see, e.g., [15]) and may be preferable in other settings.

4.6 Analysis of the Search Trajectory

To shed additional light on the behavior of the IDTS algorithm, we investigate the nature of its search trajectory. For this purpose, we carried out the following experiment on four representative instances. The algorithm was run once to solve each instance, starting from a local optimum solution obtained by the first improvement descent method. To avoid the bias of the constrained neighborhood candidate list strategy, we adopted the full swap neighborhood N_{swap}^{full} and set the maximum number of iterations to be 500.

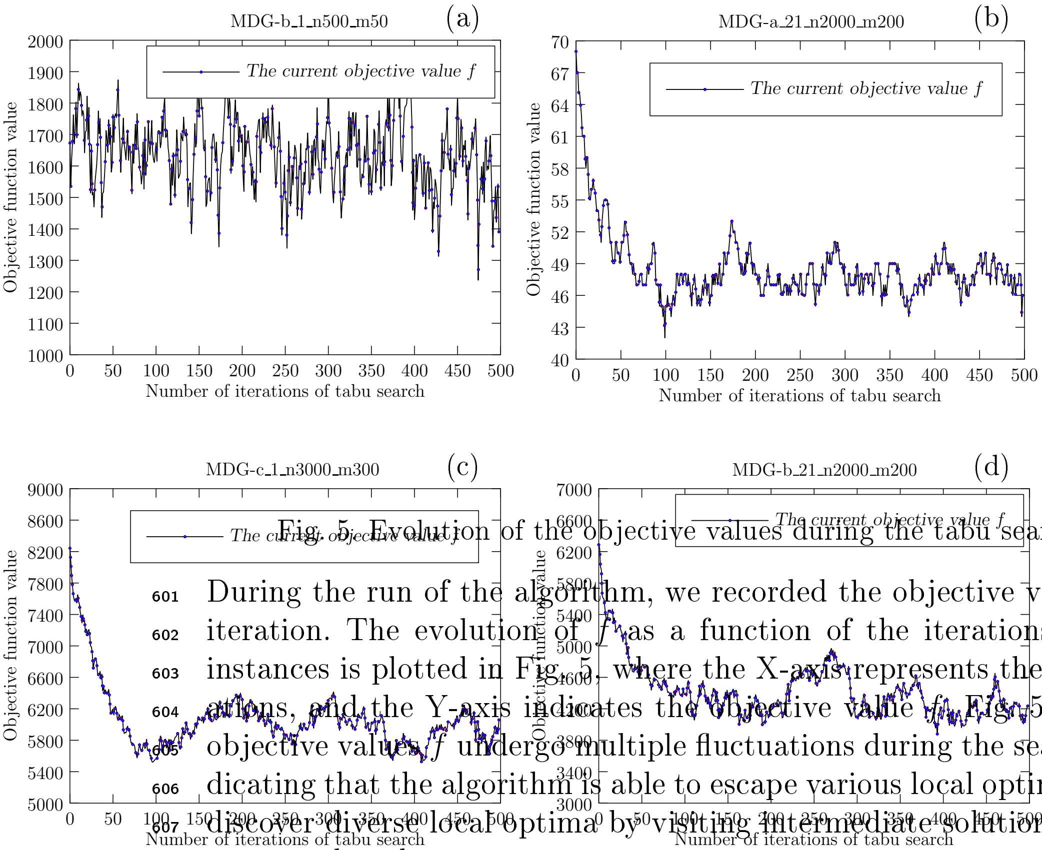

Fig. 5. Evolution of the objective values during the tabu search process.

During the run of the algorithm, we recorded the objective value (f) at each iteration. The evolution of f as a function of the iterations for the tested instances is plotted in Fig. 5, where the X-axis represents the number of iterations, and the Y-axis indicates the objective value f. Fig. 5 shows that the objective values f undergo multiple fluctuations during the search process, indicating that the algorithm is able to escape various local optimality traps and discover diverse local optima by visiting intermediate solutions whose quality can vary largely.

5 Conclusions and Future work

Our intensification-driven tabu search (IDTS) algorithm for the strongly NP-hard Min-Diff DP derives its competitive performance from three major components: a candidate list strategy utilizing a parametric reduced neighborhood to focus on promising neighbor solutions, a solution-based tabu strategy that enables a highly effective search over diverse terrain, and an intensified search mechanism that creates a refined exploration around high-quality solutions discovered during the search.

The performance of the IDTS algorithm was evaluated through extensive experiments on 250 benchmark instances commonly used to assess algorithmic performance. The computational results showed that our IDTS algorithm significantly outperforms the state-of-the-art Min-Diff DP algorithms in the literature, by finding improved best known solutions (new upper bounds) for 127 out of the 250 instances tested. Additional experiments were performed to shed light on the behavior of the proposed algorithms.

There are several possibilities to further improve our algorithm. First, self-adaptive techniques can be designed to tune the two key parameters α and θ automatically. Second, advanced diversification strategies can be investigated to better exploit the phenomenon exhibited by differential dispersion problems whereby high-quality solutions are grouped in clusters (as shown in Section 4.5). Finally, the strategies of the IDTS algorithm embody rather general principles, and it would be interesting to investigate their application more thoroughly in other binary optimization settings.

Acknowledgments

We are grateful to the reviewers for their valuable comments which helped us to improve the paper. This work was partially supported by the National Natural Science Foundation of China (Grant No. 61703213), the Natural Science Foundation of Jiangsu Province of China (Grant No. BK20170904), six talent peaks project in Jiangsu Province (Grant No. RJFW-011), and NUPTSF (Grant No. NY217154).

References

[1] Amirgaliyeva Z., Mladenovićb N., Todosijević R., Urošević D., 2017, Solving the maximum min-sum dispersion by alternating formulations of two different problems. *European Journal of Operational Research*, 260, 444–459.

[2] Aringhieri R., Cordone R., 2011, Comparing local search metaheuristics for the maximum diversity problem. *Journal of the Operational Research Society*, 62, 266–280.

[3] Aringhieri R., Cordone R., Grosso A., 2015, Construction and improvement algorithms for dispersion problems. *European Journal of Operational Research*, 242(1), 21–33.

[4] Barbati M., Piccolo C., 2016, Equality measures properties for location problems. *Optimization Letters*, 10(5), 903–920.

[5] Brimberg J., Mladenovićb N., Urošević D., Ngai E., 2009, Variable neighborhood search for the heaviest k-subgraph. *Computers & Operations Research*, 36(11), 2885–2891.

[6] Brimberg J., Mladenovićb N., Todosijević R., Urošević D., 2017, Less is more: Solving the max-mean diversity problem with variable neighborhood search. *Information Sciences*, 382,179–200.

[7] Brown J.R., 1979, The sharing problem. *Operations Research*, 27(2), 324-340.

[8] Brown J.R., 1979, The knapsack sharing problem. *Operations Research*, 27(2), 341–355.

[9] Carlton W.B., Barnes J.W., 1996, A note on hashing functions and tabu search algorithms. *European Journal of Operational Research*, 95(1), 237–239.

[10] Carlton W.B., Barnes J.W., 1996, Solving the traveling salesman problem with time windows using tabu search. *IIE Transactions*, 28, 617–629.

[11] Della Croce F., Grosso A., Locatelli M., 2009, A heuristic approach for the max-min diversity problem based on max-clique. *Computers & Operations Research*, 36(8), 2429–2433.

[12] Della Croce F., Garraffa M., Salassa F.,2016, A hybrid three-phase approach for the max-mean dispersion problem. *Computers & Operations Research*, 71, 16–22.

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

[14] Erkut E., Neuman S., 1989, Analytical models for locating undesirable facilities. *European Journal of Operational Research*, 40(3), 275–291.

[15] Glover F., Laguna. M., 1997, Tabu search. *Kluwer Academic Publishers*, Boston.

[16] Glover F., Kuo C.C., Dhir K.S., 1998, Heuristic algorithms for the maximum diversity problem. *Journal of Information and Optimization Sciences*, 19(1), 109–132.

[17] Lai X.J., Hao J.K., 2016, A tabu search based memetic search algorithm for the max-mean dispersion problem. *Computers & Operations Research*, 72, 118–127.

[18] Lai X.J., Hao J.K., 2016, Iterated maxima search for the maximally diverse grouping problem. *European Journal of Operational Research*, 254(3),780-800.

[19] Lai X.J., Yue D., Hao J.K., Glover F., 2018, Solution-based tabu search for the maximum min-sum dispersion problem. *Information Sciences*, 441, 79–94.

[20] Kerchove C., Dooren P.V., 2008, The page trust algorithm: how to rank web pages when negative links are allowed? *Proceedings SIAM International Conference on Data Mining*, 346–352.

[21] Martínez-Gavara A., Campos V., Laguna M., Martí R., 2017, Heuristic solution approaches for the maximum minsum dispersion problem. *Journal of Global Optimization*, 67(3), 671–686.

[22] Mladenović N., Todosijević R., Urošević D., 2016, Less is more: Basic variable neighborhood search for minimum differential dispersion problem. *Information Sciences*, 326, 160–171.

[23] Porumbel D.C., Hao J.K., Kuntz P, 2010, A search space cartography for guiding graph coloring heuristics. *Computers & Operations Research* 37(4): 769-778.

[24] Porumbel D.C., Hao J.K., Glover F., 2011, A simple and effective algorithm for the MaxMin diversity problem. *Annals of Operations Research*, 186(1), 275–293.

[25] Prokopyev O.A., Kong N., Martinez-Torres D.L., 2009, The equitable dispersion problem. *European Journal of Operational Research*, 197(1), 59–67.

[26] Resende M.G.C., Martí R., Gallego M., Duarte A., 2010, GRASP and path relinking for the max–min diversity problem. *Computers & Operations Research*, 37(3), 498–508.

[27] Wang Y., Wu Q., Glover F., 2017, Effective metaheuristic algorithms for the minimum differential dispersion problem. *European Journal of Operational Research*, 258, 829–843.

[28] Wu Q., Hao J.K., 2013, An adaptive multistart tabu search approach to solve the maximum clique problem. *Journal of Combinatorial Optimization*, 26(1), 86–108.

[29] Wu Q., Hao J.K., 2013, A hybrid metaheuristic method for the maximum diversity problem. *European Journal of Operational Research*, 231(2), 452–464.

[30] Woodruff D.L., Zemel E., 1993, Hashing vectors for tabu search. *Annals of Operations Research*, 41(2), 123–137.

[31] Yang B., W. Cheung, Liu J., 2007, Community mining from signed social networks. *IEEE Transactions on Knowledge & Data Engineering* 19(10), 1333–1348.

[32] Zhou Y., Hao J.K., Duval B., 2017, Opposition-based memetic search for the maximum diversity problem. *IEEE Transactions on Evolutionary Computation* 21(5), 731–745.

[33] Zhou Y., Hao J.K., 2017, An iterated local search algorithm for the minimum differential dispersion problem. *Knowledge-Based Systems* 125, 26–38.

A Appendix

We report here the results of the IDTS algorithm on the six sets of benchmarks of 170 instances that are not listed in Section 3.3. The outcomes of the computational tests are given in Tables A.1- A.6, including the previous best known results in the literature (Best Known), and for our IDTS algorithm, the best objective value (f_{best}), the average objective value (f_{avg}), the standard deviation (sdt) of objective values, and the difference between f_{best} and the Best Known results. The row 'Avg' of each table shows the average of the values in each column. The row '#Best' indicates the number of instances for which the associated result matches the current best known one, and the best results between the results of IDTS and the Best Known values are indicated in bold. In addition, the symbol '*' means that the IDTS algorithm obtained an improved solution compared to the Best Known result.

We used the same timeout limit for the IDTS algorithm as in Section 3.3, i.e., $t_{max} = n$, where n is the number of elements in the instance. The two previous studies [22,33] used the same time limit as ours. It should be noted, however, that the study in [27] set the timeout limit t_{max} according to specific instances, making it difficult to perform a direct comparison between our results and theirs on these instances. Thus, the main goal of this section is to show the detailed experimental results of our IDTS algorithm, instead of making a direct comparison between our IDTS algorithm and the algorithm in [27].

Tables A.1, A.2, and A.4 show our IDTS algorithm performed very well by comparison to the Best Known results on the MDG-a, MDG-b and GKD-c instances (which constitute all the larger instances with n = 500). Tables A.3 and A.5 show our IDTS algorithm matched or improved the Best Known results in most of GKD-b and SOM-b instances, and Table A.6 shows our algorithm yielded slightly worse outcomes compared to the Best Known results on the APOM instances. In sum, these computational results further show a good search ability of the proposed IDTS algorithm.

Table A.1
Computational results on MDG-a instances with $n = 500$.

Instance	Time (s)	Best known	f_{best}	f_{avg}	std	Δf_{best}
MDG-a_1_n500_m50	500	10.46	**9.73***	10.97	0.37	-0.73
MDG-a_2_n500_m50	500	10.58	**10.21***	11.00	0.40	-0.37
MDG-a_3_n500_m50	500	10.74	**10.04***	11.03	0.32	-0.70
MDG-a_4_n500_m50	500	10.90	**10.10***	10.99	0.36	-0.80
MDG-a_5_n500_m50	500	10.58	**10.02***	10.97	0.35	-0.56
MDG-a_6_n500_m50	500	10.08	**9.91***	10.99	0.41	-0.17
MDG-a_7_n500_m50	500	10.35	**9.55***	11.07	0.44	-0.80
MDG-a_8_n500_m50	500	**10.16**	10.35	10.92	0.35	0.19
MDG-a_9_n500_m50	500	**9.97**	10.47	11.06	0.28	0.50
MDG-a_10_n500_m50	500	10.58	**10.52***	11.10	0.31	-0.06
MDG-a_11_n500_m50	500	10.57	**9.37***	10.95	0.43	-1.20
MDG-a_12_n500_m50	500	10.62	**10.17***	11.11	0.30	-0.45
MDG-a_13_n500_m50	500	**10.31**	10.32	11.16	0.30	0.01
MDG-a_14_n500_m50	500	**9.95**	9.96	10.99	0.34	0.01
MDG-a_15_n500_m50	500	10.40	**9.66***	11.01	0.38	-0.74
MDG-a_16_n500_m50	500	10.40	**10.28***	10.92	0.29	-0.12
MDG-a_17_n500_m50	500	**10.33**	10.34	11.02	0.33	0.01
MDG-a_18_n500_m50	500	10.56	**10.16***	10.95	0.29	-0.40
MDG-a_19_n500_m50	500	10.46	**9.55***	10.88	0.41	-0.91
MDG-a_20_n500_m50	500	10.54	**9.96***	11.03	0.39	-0.58
Avg		10.43	**10.03**	11.01	0.35	-0.39
#Best		5	15			

Table A.2
Computational results on MDG-b instances with $n = 500$.

Instance	Time (s)	Best known	f_{best}	f_{avg}	std	Δf_{best}
MDG-b_1_n500_m50	500	1055.33	**1031.91***	1120.95	33.23	-23.42
MDG-b_2_n500_m50	500	1038.08	**993.71***	1112.43	37.34	-44.37
MDG-b_3_n500_m50	500	1086.91	**1045.74***	1118.47	32.95	-41.17
MDG-b_4_n500_m50	500	1052.27	**944.13***	1097.53	38.75	-108.14
MDG-b_5_n500_m50	500	**1005.45**	1013.51	1104.18	38.26	8.06
MDG-b_6_n500_m50	500	1061.50	**1002.18***	1107.08	39.33	-59.32
MDG-b_7_n500_m50	500	1063.67	**937.19***	1099.44	41.89	-126.48
MDG-b_8_n500_m50	500	1088.63	**1026.35***	1120.24	30.60	-62.28
MDG-b_9_n500_m50	500	1069.26	**1047.74***	1115.17	35.46	-21.52
MDG-b_10_n500_m50	500	1069.54	**1006.26***	1114.27	39.39	-63.28
MDG-b_11_n500_m50	500	**1031.02**	1047.57	1121.52	33.07	16.55
MDG-b_12_n500_m50	500	1063.76	**1011.66***	1107.38	38.17	-52.10
MDG-b_13_n500_m50	500	1026.86	**990.38***	1106.17	43.44	-36.48
MDG-b_14_n500_m50	500	**1018.69**	1062.11	1120.50	29.36	43.42
MDG-b_15_n500_m50	500	**1022.19**	1044.68	1115.20	28.77	22.49
MDG-b_16_n500_m50	500	1057.20	**1035.26***	1112.72	28.83	-21.94
MDG-b_17_n500_m50	500	1045.20	**1041.10***	1120.33	31.46	-4.10
MDG-b_18_n500_m50	500	1032.54	**998.27***	1095.49	39.46	-34.27
MDG-b_19_n500_m50	500	1066.78	**982.59***	1089.50	38.66	-84.19
MDG-b_20_n500_m50	500	1022.66	**1013.54***	1102.86	37.12	-9.12
Avg	500	1048.88	**1013.79**	1110.07	35.78	-35.08
#Best		4	16			

Table A.3
Computational results on GKD-b instances.

Instance	Time (s)	Best known	f_{best}	f_{avg}	std	Δf_{best}
GKD-b_1_n25_m2	25	**0.00**	**0.00**	0.00	0.00	0.00
GKD-b_2_n25_m2	25	**0.00**	**0.00**	0.00	0.00	0.00
GKD-b_3_n25_m2	25	**0.00**	**0.00**	0.00	0.00	0.00
GKD-b_4_n25_m2	25	**0.00**	**0.00**	0.00	0.00	0.00
GKD-b_5_n25_m2	25	**0.00**	**0.00**	0.00	0.00	0.00
GKD-b_6_n25_m7	25	**12.72**	**12.72**	12.72	0.00	0.00
GKD-b_7_n25_m7	25	**14.10**	**14.10**	14.10	0.00	0.00
GKD-b_8_n25_m7	25	**16.76**	**16.76**	16.76	0.00	0.00
GKD-b_9_n25_m7	25	**17.07**	**17.07**	17.07	0.00	0.00
GKD-b_10_n25_m7	25	**23.27**	**23.27**	23.86	1.19	0.00
GKD-b_11_n50_m5	50	**1.93**	**1.93**	1.93	0.00	0.00
GKD-b_12_n50_m5	50	**2.05**	**2.05**	2.05	0.01	0.00
GKD-b_13_n50_m5	50	**2.36**	**2.36**	2.43	0.22	0.00
GKD-b_14_n50_m5	50	**1.66**	**1.66**	1.66	0.00	0.00
GKD-b_15_n50_m5	50	**2.85**	**2.85**	2.85	0.00	0.00
GKD-b_16_n50_m15	50	**42.75**	**42.75**	42.93	0.66	0.00
GKD-b_17_n50_m15	50	**48.11**	**48.11**	50.54	7.29	0.00
GKD-b_18_n50_m15	50	**43.20**	**43.20**	43.20	0.00	0.00
GKD-b_19_n50_m15	50	**46.41**	**46.41**	46.41	0.00	0.00
GKD-b_20_n50_m15	50	**47.72**	**47.72**	48.25	1.92	0.00
GKD-b_21_n100_m10	100	**9.33**	**9.33**	11.47	1.26	0.00
GKD-b_22_n100_m10	100	**8.60**	**8.60**	12.16	1.34	0.00
GKD-b_23_n100_m10	100	**6.91**	7.59	10.52	1.53	0.68
GKD-b_24_n100_m10	100	**7.59**	**7.59**	11.85	1.69	0.00
GKD-b_25_n100_m10	100	**6.91**	9.64	12.04	1.19	2.73
GKD-b_26_n100_m30	100	**159.19**	**159.19**	162.64	6.99	0.00
GKD-b_27_n100_m30	100	**124.17**	**124.17**	141.46	24.47	0.00
GKD-b_28_n100_m30	100	**106.38**	**106.38**	119.41	16.86	0.00
GKD-b_29_n100_m30	100	**135.85**	**135.85**	138.53	7.47	0.00
GKD-b_30_n100_m30	100	**127.27**	**127.27**	136.05	13.51	0.00
GKD-b_31_n125_m12	125	**11.05**	**11.05**	12.80	2.05	0.00
GKD-b_32_n125_m12	125	11.43	**10.43***	14.85	1.47	-1.00
GKD-b_33_n125_m12	125	**9.18**	10.79	13.93	1.40	1.61
GKD-b_34_n125_m12	125	**11.83**	**11.83**	16.22	1.63	0.00
GKD-b_35_n125_m12	125	9.20	**7.53***	11.88	1.60	-1.67
GKD-b_36_n125_m37	125	**125.55**	**125.55**	146.88	17.19	0.00
GKD-b_37_n125_m37	125	**194.22**	**194.22**	194.65	1.53	0.00
GKD-b_38_n125_m37	125	**184.27**	**184.27**	190.89	17.66	0.00
GKD-b_39_n125_m37	125	**155.39**	**155.39**	161.74	6.29	0.00
GKD-b_40_n125_m37	125	**161.68**	172.80	199.71	11.79	11.12
GKD-b_41_n150_m15	150	**16.48**	17.85	22.22	1.85	1.37
GKD-b_42_n150_m15	150	**12.38**	**12.38**	20.03	2.67	0.00
GKD-b_43_n150_m15	150	**11.83**	13.99	18.42	1.84	2.16
GKD-b_44_n150_m15	150	16.58	**11.74***	18.20	2.33	-4.84
GKD-b_45_n150_m15	150	16.43	**12.84***	19.95	2.24	-3.59
GKD-b_46_n150_m45	150	**207.81**	**207.81**	219.40	7.26	0.00
GKD-b_47_n150_m45	150	**211.77**	**211.77**	214.20	5.74	0.00
GKD-b_48_n150_m45	150	**177.29**	**177.29**	203.37	17.70	0.00
GKD-b_49_n150_m45	150	**197.88**	**197.88**	204.88	10.73	0.00
GKD-b_50_n150_m45	150	**220.76**	230.49	246.24	23.38	9.73
Avg		**59.56**	59.93	64.67	4.52	0.37
#Best		46	43			

Table A.4
Computational results on GKD-c instances.

Instance	Time (s)	Best known	f_{best}	f_{avg}	std	Δf_{best}
GKD-c_1_n500_m50	500	**6.39**	6.51	7.93	0.93	0.12
GKD-c_2_n500_m50	500	**6.13**	6.75	8.34	0.84	0.62
GKD-c_3_n500_m50	500	6.65	**6.10***	8.29	0.93	-0.55
GKD-c_4_n500_m50	500	6.64	**5.59***	7.97	1.06	-1.05
GKD-c_5_n500_m50	500	7.38	**6.88***	8.70	1.11	-0.50
GKD-c_6_n500_m50	500	6.79	**6.29***	7.87	0.93	-0.50
GKD-c_7_n500_m50	500	**6.84**	7.11	8.88	1.02	0.27
GKD-c_8_n500_m50	500	**7.01**	7.27	9.16	1.31	0.26
GKD-c_9_n500_m50	500	8.09	**6.18***	8.31	0.97	-1.91
GKD-c_10_n500_m50	500	7.37	**6.85***	9.27	1.04	-0.52
GKD-c_11_n500_m50	500	6.42	**5.27***	7.73	1.04	-1.15
GKD-c_12_n500_m50	500	6.50	**6.12***	8.14	1.02	-0.38
GKD-c_13_n500_m50	500	**6.52**	7.27	8.82	1.24	0.75
GKD-c_14_n500_m50	500	6.38	**5.98***	8.43	1.11	-0.40
GKD-c_15_n500_m50	500	6.99	**6.32***	8.47	1.04	-0.67
GKD-c_16_n500_m50	500	6.51	**5.88***	7.91	1.18	-0.63
GKD-c_17_n500_m50	500	6.31	**5.62***	7.50	1.06	-0.69
GKD-c_18_n500_m50	500	6.88	**6.51***	8.61	0.97	-0.37
GKD-c_19_n500_m50	500	6.84	**6.20***	8.26	1.11	-0.64
GKD-c_20_n500_m50	500	6.32	**5.53***	8.10	1.17	-0.79
Avg		6.75	**6.31**	8.33	1.05	-0.44
#Best		5	15			

Table A.5
Computational results on SOM-b instances.

Instance	Time (s)	Best known	f_{best}	f_{avg}	std	Δf_{best}
SOM-b_1_n100_m10	100	**0**	**0**	1.4	0.49	0
SOM-b_2_n100_m20	100	**4**	**4**	5.15	0.36	0
SOM-b_3_n100_m30	100	**6**	7	8.25	0.54	1
SOM-b_4_n100_m40	100	**10**	**10**	11.2	0.68	0
SOM-b_5_n200_m20	200	**3**	**3**	4.55	0.5	0
SOM-b_6_n200_m40	200	**9**	**9**	9.85	0.36	0
SOM-b_7_n200_m60	200	**13**	**13**	14.55	0.67	0
SOM-b_8_n200_m80	200	**18**	**18**	19.65	0.91	0
SOM-b_9_n300_m30	300	**6**	**6**	6.85	0.36	0
SOM-b_10_n300_m60	300	**12**	**12**	13.4	0.49	0
SOM-b_11_n300_m90	300	**18**	**18**	19.5	0.74	0
SOM-b_12_n300_m120	300	24	**23***	25.85	1.19	-1
SOM-b_13_n400_m40	400	9	**8***	8.95	0.22	-1
SOM-b_14_n400_m80	400	**16**	**16**	17.15	0.61	0
SOM-b_15_n400_m120	400	**23**	**23**	24.4	0.86	0
SOM-b_16_n400_m160	400	**27**	30	32.55	1.28	3
SOM-b_17_n500_m50	500	**10**	**10**	10.7	0.64	0
SOM-b_18_n500_m100	500	**19**	**19**	20.2	0.51	0
SOM-b_19_n500_m150	500	**26**	**26**	28.75	1.3	0
SOM-b_20_n500_m200	500	**34**	36	39.45	2.48	2
Avg	300	**14.35**	14.55	16.12	0.76	0.2
#Best		18	17			

Table A.6
Computational results on APOM instances.

Instance	Time (s)	Best known	f_{best}	f_{avg}	std	Δf_{best}
01a050m10	50	**1.41**	**1.41**	1.87	0.16	0.00
02a050m20	50	**14.72**	**14.72**	14.73	0.06	0.00
03a100m20	100	**3.65**	4.01	4.38	0.32	0.36
04a100m40	100	**25.50**	**25.50**	26.42	2.11	0.00
05a150m30	150	**6.56**	7.09	7.91	0.72	0.53
06a150m60	150	**46.99**	**46.99**	47.31	0.79	0.00
07a200m40	200	**11.39**	11.49	12.46	0.83	0.10
08a200m80	200	63.48	**63.46***	64.47	1.94	-0.02
09a250m50	250	**14.56**	14.68	16.61	1.18	0.12
10a250m100	250	**82.09**	82.51	86.04	4.78	0.43
11b050m10	50	**1091.00**	1355.00	2043.30	326.29	264.00
12b050m20	50	**5552.00**	**5552.00**	6044.15	370.60	0.00
13b100m20	100	**3996.00**	4160.00	4945.20	406.45	164.00
14b100m40	100	**9540.00**	10552.00	11360.45	357.56	1012.00
15b150m30	150	6769.00	**6607.00***	7386.60	437.72	-162.00
16b150m60	150	**13449.00**	14007.00	15101.85	533.94	558.00
17b200m40	200	**8197.00**	9042.00	9809.65	361.10	845.00
18b200m80	200	**17502.00**	18026.00	19085.30	479.00	524.00
19b250m50	250	11427.00	**10635.00***	11730.05	447.96	-792.00
20b250m100	250	21832.00	**20963.00***	22197.45	754.33	-869.00
21c050m10	50	1149.00	**1124.00**	1225.70	100.52	-25.00
22c050m20	50	**6205.00**	**6205.00**	6210.80	25.28	0.00
23c100m20	100	2239.00	**2149.00***	2850.05	299.25	-90.00
24c100m40	100	**11098.00**	**11098.00**	13278.50	5263.04	0.00
25c150m30	150	3550.00	**3414.00***	4757.40	1705.96	-136.00
26c150m60	150	**13087.00**	**13087.00**	21426.80	14445.11	0.00
27c200m40	200	**4865.00**	5226.00	8445.60	3238.32	361.00
28c200m80	200	**19393.00**	19537.00	26525.50	20460.89	144.00
29c250m50	250	**5650.00**	5955.00	10390.00	3572.99	305.00
30c250m100	250	**22050.00**	22280.00	34583.35	16810.51	230.00
31d050m10	50	**1049.00**	**1049.00**	1138.85	102.52	0.00
32d050m20	50	**4564.00**	**4564.00**	4587.15	100.91	0.00
33d100m20	100	**2374.00**	2561.00	2847.45	176.55	187.00
34d100m40	100	**8979.00**	**8979.00**	13011.00	7666.21	0.00
35d150m30	150	**3234.00**	3923.00	6545.45	2148.50	689.00
36d150m60	150	**12444.00**	**12444.00**	15813.80	6053.84	0.00
37d200m40	200	**4752.00**	5113.00	8731.80	2839.81	361.00
38d200m80	200	**18683.00**	18835.00	23145.80	8027.08	152.00
39d250m50	250	**5856.00**	6142.00	11381.45	3598.45	286.00
40d250m100	250	**21001.00**	21492.00	46862.40	41716.38	491.00
Avg.	150	**6796.18**	6908.70	9343.63	3571.00	112.51
#Best		33	19			