Neighborhood decomposition based variable neighborhood search and tabu search for maximally diverse grouping

Xiangjing Lai [a], Jin-Kao Hao [b,c,*], Zhang-Hua Fu [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] *Shenzhen Institute of Artificial Intelligence and Robotics for Society, and The Chinese University of Hong Kong, Shenzhen, Shenzhen 518172, China*

Accepted to *European Journal of Operational Research*, July 2020

Abstract

The maximally diverse grouping problem (MDGP) is a relevant NP-hard optimization problem with a number of real-world applications. However, solving large instances of the problem is computationally challenging. This work is dedicated to a new heuristic algorithm for the problem, which distinguishes itself by two original features. First, it introduces the first neighborhood decomposition strategy to accelerate neighborhood examinations. Second, it integrates, in a probabilistic way, two complementary neighborhood decomposition based local search procedures (variable neighborhood descent and tabu search) as well as an adaptive perturbation strategy to ensure a suitable balance between intensification and diversification of the search space. Computational results on 320 benchmark instances commonly used in the literature show that the proposed algorithm competes favorably with the state-of-the-art MDGP algorithms, by reporting improved best-known results (new lower bounds) of the literature for 220 large instances. Additional experiments are conducted to analyze the main components of the algorithm. The proposed algorithm can help to better solve practical problems that can be formulated by the maximally diverse grouping model.

Keywords: Heuristics; grouping and clustering; neighborhood decomposition; local search.

* Corresponding author.

Email addresses: laixiangjing@gmail.com (Xiangjing Lai), jin-kao.hao@univ-angers.fr (Jin-Kao Hao), fuzhanghua@cuhk.edu.cn

1 Introduction

Given a set V of N elements, a distance matrix $D = [d_{ij}]_{N\times N}$ between the elements, a positive integer m, and m pairs of non-negative integers $\{L_g, U_g\}$ $(1 \le g \le m)$ called the capacity lower and upper limits of groups, the maximally diverse grouping problem (MDGP) is to partition the set V into m disjoint groups such that the size of each group g lies in $[L_g, U_g]$ $(1 \le g \le m)$, while the sum of the distances between the elements in the same groups is maximized. MDGP can also be described as a graph partition problem as follows. We consider an edge-weighted complete graph $G = (V, E, D)$, where V is the set of N vertices, E is the set of $N\times(N-1)/2$ edges, and $D = [d_{ij}]_{N\times N}$ defines the set of edge weights. Then MDGP can be considered as a special case of the NP-hard clique partitioning problem (CPP) [13,29] with non-negative edge weights and constraints related to the capacity lower and upper limits of groups.

Formally, MDGP can be written as a quadratic binary programming problem [11,23]:

$$\text{Maximize} \quad \sum_{g=1}^{m}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} d_{ij}X_{ig}X_{jg} \tag{1}$$

$$\text{Subject to} \quad \sum_{g=1}^{m} X_{ig} = 1, i = 1, 2, \ldots, N \tag{2}$$

$$L_g \le \sum_{i=1}^{N} X_{ig} \le U_g, g = 1, 2, \ldots, m \tag{3}$$

$$X_{ig} \in \{0, 1\}, \forall i \in \{1, 2, \ldots, N\}, \forall g \in \{1, 2, \ldots, m\}, \tag{4}$$

where X_{ig} is a binary variable that takes 1 if the vertex i locates in the group g and 0 otherwise, the set of constraints (2) guarantees that each vertex is located in exactly one group, and the set of constraints (3) ensures that the size of group g lies in $[L_g, U_g]$ $(g = 1, 2, \ldots, m)$.

MDGP is a relevant model for formulating many practical problems, such as assignment of students to groups [16,17,28], creation of peer review groups [5], and VLSI design [26]. More applications can be found in [11,18,22,23,25].

Due to the NP-hardness of MDGP [10], a number of heuristic algorithms have been proposed to find approximate solutions. Existing heuristic algorithms can be classified into two categories, i.e., trajectory-based local search algorithms and hybrid evolutionary algorithms. As examples of trajectory-based local

(Zhang-Hua Fu), medongy@vip.163.com (Dong Yue).

search algorithms, we mention the multistart algorithm [1], Lotfi-Cerveny-Weitz (LCW) algorithm [27], T-LCW method mixing LCW and tabu search [11], simulated annealing algorithm [21], variable neighborhood search algorithms [4,21,25], tabu search algorithms [11,22], and iterated maxima search (IMS) algorithm [18]. Hybrid evolutionary algorithms include hybrid genetic algorithms [9,24], hybrid grouping genetic algorithm [5], hybrid steady-state genetic algorithm [21], artificial bee colony (ABC) algorithms [23], and constructive genetic algorithm [19]. According to the experimental results reported in recent studies such as [4,18,24], the skewed general variable neighborhood search algorithm (SGVNS) [4], the iterated tabu search algorithm (ITS) [22], the iterated maxima search algorithm (IMS) [18] can be regarded as the current state-of-the-art MDGP algorithms. Finally, the hybrid genetic algorithm NSGGA [24] can also be considered as a state-of-the-art algorithm, but only for the special case where all groups have an equal size.

One notices that the best performing MDGP algorithms in the literature rely, with no exception, on a powerful neighborhood search subroutine. Meanwhile, a careful analysis of the underlying neighborhood search procedures indicates that they examine the whole neighborhood used at each iteration, which leads to superfluous examinations of many non-promising neighbor solutions and a waste of computation time. To overcome this problem, this work investigates an original neighborhood decomposition method which avoids redundant calculations of the neighborhood examination and thus speeds up the neighborhood search.

The main contributions of this work are summarized as follows.

(1) We propose a novel neighborhood decomposition based heuristic algorithm (NDHA) with two original features. First, NDHA relies a dynamic neighborhood decomposition strategy that allows the algorithm to avoid redundant examination of irrelevant neighbor solutions at each iteration by ignoring uninteresting candidate solutions. As such, the neighborhood decomposition strategy accelerates the neighborhood examination and enables more promising candidate solutions to be examined for a given time budget. Second, the neighborhood decomposition based heuristic algorithm integrates two complementary neighborhood search procedures (i.e., tabu search and variable neighborhood descent) that are applied in a probabilistic way, leading to an enhanced robustness of the algorithm.
(2) We present computational results of the proposed algorithm on 320 benchmark instances commonly used in the literature and compare our results with those of the state-of-the-art MDGP algorithms. Our comparative studies indicate that the proposed algorithm outperforms significantly the reference algorithms especially on the large benchmark instances. The new lower bounds for 220 benchmark instances reported by our algorithm are useful for assessment of other MDGP algorithms. Moreover,

the source code of our NDHA algorithm will be made available online, which can be used by researchers and practitioners to better solve various practical problems that can be formulated as MDGP.

(3) The neighborhood decomposition strategy is of general nature and can be advantageously adopted to speed up other neighborhood search algorithms for MDGP and other related clustering problems as well.

In the next section, we describe the proposed algorithm. In Section 3, we evaluate the proposed algorithm by reporting computational results on a large number of benchmark instances and making comparisons with reference algorithms in the literature. In Section 4, we conduct analyses to investigate two essential components of the proposed algorithm as well as one key parameter. In the last section, we summarize the work and provide research perspectives.

2 Neighborhood decomposition based hybrid heuristic for MDGP

The neighborhood decomposition based heuristic algorithm (NDHA) proposed in this work follows the general iterated local search framework [20] and combines two complementary neighborhood search procedures (i.e., tabu search and variable neighborhood descent) with a perturbation operator to reach a suitable tradeoff between intensification and diversification of the search space. Compared to existing algorithms, the proposed algorithm distinguishes itself by two key features, i.e., its neighborhood decomposition strategy aiming to speed up the neighborhood search and a combined use of two local search methods aiming to enhance the robustness of the algorithm. We describe the main framework of the proposed algorithm and its components in this section.

2.1 Main Framework

The NDHA algorithm (Algorithm 1) starts with the solution initialization procedure (Section 2.3) to obtain a high-quality initial feasible solution. It then performs a number of iterations to improve the current solution until the given time limit (t_{max}) is reached (lines 4–26).

At each iteration, the current solution s is first perturbed by the perturbation operator (line 5, Section 2.5) and the perturbed solution is then improved by the neighborhood decomposition tabu search (NDTS) procedure (Section 2.4.4) or the neighborhood decomposition variable neighborhood descent (NDVND) procedure (Section 2.4.3). The decision of applying NDTS or NDVND depends on a probability: $Q \times \frac{m}{N}$ for NDTS and $1 - (Q \times \frac{m}{N})$ for NDVND, where Q ($0 \leq Q \leq N/m$) is a parameter, N is the number of elements in the

Algorithm 1: Neighborhood decomposition based heuristic algorithm (NDHA) for MDGP

Input: An edge-weighted complete graph $G = (V, E, D)$, number of groups m, time limit t_{max}, and parameters δ, Q, k_{min}, k_{step}, k_{max}
Output: The best feasible solution found (s^*)

1 $s \leftarrow InitialSolution(G, m)$ `/* Section 2.3 */`
2 $s^* \leftarrow s$ `/*` s^* `records the best solution found */`
3 $k \leftarrow k_{min}$ `/*` k `denotes the current perturbation strength */`
4 **while** $Time() \leq t_{max}$ **do**
5     $s' \leftarrow Perturbation(s, k)$ `/* Algorithm 4 */`
6     $r \leftarrow rand(0, 1)$ `/*` $rand(0, 1)$ `denotes a random number in` $(0, 1)$ `*/`
7   **if** $r < Q \times \frac{m}{N}$ **then**
8         $s'' \leftarrow NDTS(s')$ `/* Tabu search, Algorithm 3 */`
9   **end**
10   **else**
11         $s'' \leftarrow NDVND(s')$ `/* Variable neighborhood descent, Algorithm 2 */`
12   **end**
13   **if** $(\frac{f(s'')}{f(s)} + \delta \cdot d(s'', s) > 1) \wedge (\frac{f(s'')}{f(s^*)} + \delta \cdot d(s'', s^*) > 1$ **then**
14         $s \leftarrow s''$
15   **end**
16   **if** $f(s'') > f(s^*)$ **then**
17         $s^* \leftarrow s''$
18         $k \leftarrow k_{min}$
19   **end**
20   **else**
21         $k \leftarrow k + k_{step}$
22   **end**
23   **if** $k \geq k_{max}$ **then**
24         $k \leftarrow k_{min}$
25   **end**
26 **end**
27 **return** s^*

problem instance, and m is the number of groups. Subsequently, the improved solution s'' is accepted, like [4], as the current solution based on its objective value $f(s'')$ and its distances to the current solution s and the best solution found so far s^* (lines 13–15), where the distances between solutions are measured by a partition-based distance function from [4,18]. After that, the best solution found so far (s^*) is accordingly updated if s'' is better than s^* (lines 16–19).

The strength k of the perturbation operator is adaptively adjusted during the search process following the strategies of breakout local search [2,3]. k is initially set to the minimum value k_{min} (line 3). Then, k increases by k_{step} if the recorded best solution s^* is not updated by s'' and is reset to k_{min} otherwise (lines 16–22). When k reaches the maximum value k_{max}, it is reset to k_{min} as well (lines 23–25). In the following subsections, we describe the components of the NDHA algorithm.

2.2 Search Space and Solution Representation

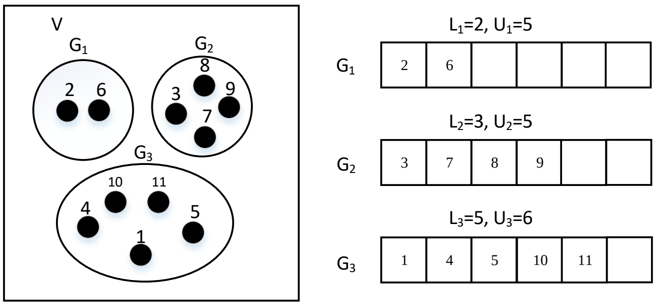


Fig. 1. An illustrative example for solution representation with $N = 11$, $m = 3$, and $U_{max} = 6$. We use a 3×6 matrix A (right figure) to indicate a solution (left figure).

Given an edge-weighted complete graph $G = (V, E, D)$ and m pairs of capacity limits of groups $\{L_g, U_g\}$ ($L_g \leq U_g$, $g = 1, 2, \ldots, m$), the search space Ω explored by the NDHA algorithm contains all m-partitions $\{G_1, G_2, \ldots, G_m\}$ of the vertex set V satisfying the capacity limits of groups, i.e., $G_1 \cup G_2 \cup \cdots \cup G_m = \mathrm{V}$, $G_i \cap G_j = \emptyset$ ($i \neq j$), and $L_g \leq |G_g| \leq U_g$ for $g = 1, 2, \ldots, m$.

To ensure a high computational efficiency, we use a N-dimensional vector $x[1:N]$ to represent a candidate solution (i.e., a m-partition of V), where $x[i]$ ($i = 1, 2, \ldots, N$) takes its values in $\{1, 2, \ldots, m\}$ and $x[i] = g$ indicates that vertex i is clustered in group g. In addition, to ease the implementation of the neighborhood decomposition strategy, we also maintain a $m \times U_{max}$ matrix A, as illustrated in Fig. 1, where U_{max} denotes the largest upper limit of groups, i.e., $U_{max} = max_{1 \leq g \leq m}\{U_g\}$.

2.3 Initial Solution

Following [18], the initialization procedure of the NDHA algorithm generates β feasible solutions (typically from ten to a few dozen) and then chooses the best one among them as the initial solution of the algorithm. Specifically, the initialization procedure generates a feasible solution as follows. First, it constructs randomly a partial solution satisfying the lower capacity limits of groups, i.e., $|G_g| \geq L_g$ ($g = 1, 2, \ldots, m$). Then, the remaining vertices are added into the partial solution one by one to obtain a complete feasible solution such that the upper capacity limits of groups are respected, i.e., $|G_g| \leq U_g$ $(g = 1, 2, \ldots, m)$. Finally, the quality of the constructed solution is locally improved by the local search procedure described in Algorithm 2.

2.4 Neighborhood Decomposition based Local Search Procedures

The NDHA algorithm relies on two key local search procedures which are used to improve solutions generated by the initialization procedure or the perturbation operator. We describe below these two local search procedures.

2.4.1 Neighborhood Structures and their Decomposition

The local search procedures of the NDHA algorithm are based the neighborhood decomposition strategy introduced in this work. Below, we present the two underlying neighborhoods used in this work (the constrained $OneMove$ neighborhood and swap neighborhood) and their decompositions. Though both neighborhoods have been used in existing studies on MDGP [4,11,18,21–23,25], the neighborhood decomposition strategy is novel, which constitutes one of the key ingredients contributing to the success of our NDHA algorithm.

The constrained $OneMove$ neighborhood (denoted by N_1) can be described as follows. Given a solution $s = \{G_1, G_2, \ldots, G_m\}$, the $OneMove$ move denoted by (v, i, j) transfers a vertex v from its current group G_i to another group G_j $(j \neq i)$ such that the resulting neighbor solution (denoted by $s \oplus (v, i, j)$) satisfies the capacity constraints of groups G_i and G_j. The neighborhood $N_1(s)$ of solution s is then composed of all feasible solutions which can be obtained by applying $OneMove$ to s: $N_1(s) = \{s \oplus (v, i, j) : v \in G_i, i, j \in \{1, \ldots, m\},\ i \neq j, |G_i| > L_i, |G_j| < U_j\}$. Clearly $N_1(s)$ has a size bounded by $O(N \times m)$, and becomes empty if $L_g = U_g$ for any group g.

We notice that $N_1(s)$ can be decomposed into $m \times (m-1)$ disjoint subsets that we call *neighborhood blocks* $B_1[i][j](s)$ $(i, j \in \{1, \ldots, m\},\ i \neq j)$, where each neighborhood block $B_1[i][j](s)$ is given by $B_1[i][j](s) = \{s \oplus (v, i, j) : v \in G_i, |G_i| > L_i, |G_j| < U_j\}$. As a result, the neighborhood $N_1(s)$ can be equivalently expressed as $N_1(s) = \cup_{1 \leq i \neq j \leq m} B_1[i][j](s)$.

Now we can use this decomposed neighborhood to accelerate neighborhood examination as follow. At each iteration of the neighborhood search, the current neighborhood N_1 is first decomposed into $m \times (m-1)$ blocks. Then the algorithm skips those non-promising blocks that have been identified in previous iterations and marked in a state matrix (see below), and focuses only on the remaining (promising) blocks. As a result, the neighborhood search is speeded up greatly.

To indicate whether a neighborhood block has been checked or not during the neighborhood search process, we maintain a $m \times m$ asymmetric binary state matrix M_1, where the entry $M_1[i][j]$ $(1 \leq i \neq j \leq m)$ corresponds to the neighborhood block $B_1[i][j](s)$ and the diagonal entries $M_1[i][i]$ $(1 \leq i \leq m)$ are irrelevant since the corresponding $B_1[i][i](s)$ $(1 \leq i \leq m)$ are empty set. $M_1[i][j]$ takes 0 if the block $B_1[i][j](s)$ has been examined previously without finding any improving solution, and takes 1 otherwise. An illustrative example for the state matrix M_1 is shown in Fig. 2(a).

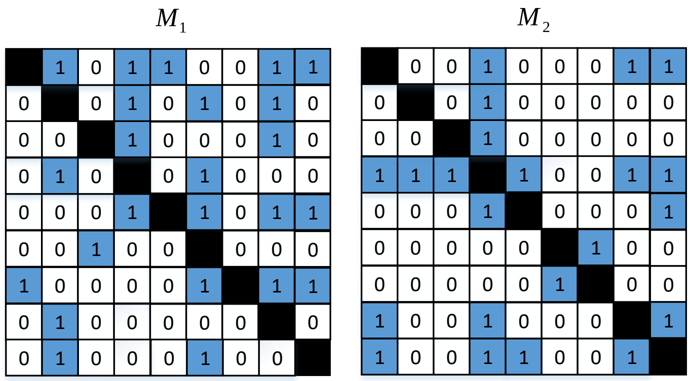
The swap neighborhood (denoted by N_2) is defined by the $Swap(v, u)$ move which generates a neighbor solution by exchanging the group of vertex v and the group of vertex u. Therefore, the neighborhood $N_2(s)$ of s contains all possible solutions that can be reached by applying the $Swap$ move to s: $N_2(s) = \{s \oplus Swap(v, u) : v \in G_i, u \in G_j, 1 \leq i < j \leq m\}$. This neighborhood has a size bounded by $O(N^2)$.

Similar to N_1, the neighborhood $N_2(s)$ can also be decomposed into $m \times (m-1)/2$ disjoint neighborhood blocks, i.e., $N_2(s) = \cup_{1 \leq i < j \leq m} B_2[i][j](s)$, where each neighborhood block $B_2[i][j](s)$ is defined by $B_2[i][j](s) = \{s \oplus Swap(v, u) : v \in G_i, u \in G_j\}$. Then, taking advantage of this decomposed neighborhood, we can speed up neighborhood examination during the search process by ignoring the non-promising neighborhood blocks that do not contain any improving solution. For this purpose, we use a $m \times m$ symmetric binary state matrix M_2, where the entry $M_2[i][j]$ $(= M_2[j][i])$ corresponds to the block $B_2[i][j](s)$ and takes 0 if $B_2[i][j](s)$ has been examined previously without finding an improving solution, and takes 1 otherwise. An illustrative example of M_2 is given in Fig. 2(b).

The neighborhood decomposition method is based on the fact that the objective function of MDGP is the sum of subunit objectives defined on m groups and thus most neighborhood blocks are mutually independent in terms of move values (i.e., the change of objective values between the current solution and a neighbor solution). With the help of the state matrix of the corresponding neighborhood, the algorithm avoids many redundant neighborhood examinations if the neighborhood is checked in a block-by-block way, as described in Sections 2.4.3 and 2.4.4.

2.4.2 Updating of State Matrices of Neighborhoods

The state matrices M_1 and M_2 are initialized at the beginning of each neighborhood search procedure, where their entries are set to 1 except for the diagonal elements which are set definitively to 0. Then, M_1 and M_2 are dynamically updated as the search process progresses.



(a) state matrix of N_1 (b) state matrix of N_2

Fig. 2. State matrices of the constrained *OneMove* neighborhood N_1 and the swap neighborhood N_2, where M_1 and M_2 both are a $m \times m$ ($m = 9$) 0–1 matrix. For the current solution s, the entries $M_1[i][j]$ and $M_2[i][j]$ correspond respectively to the neighborhood blocks $B_1[i][j](s)$ and $B_2[i][j](s)$, which take 0 if the corresponding block has been checked without finding an improving solution.

Specifically, for the neighborhood N_1, $M_1[i][j]$ is first set to 0 when the block $B_1[i][j](s)$ is being examined. For N_2, both $M_2[i][j]$ and $M_2[j][i]$ are set to 0 when the block $B_2[i][j](s)$ is being examined. After that, $M_1[i][q]$ (or $M_2[i][q]$ for N_2), $M_1[q][i]$ (or $M_2[q][i]$ for N_2), $M_1[j][q]$ (or $M_2[j][q]$ for N_2) and $M_1[q][j]$ (or $M_2[q][j]$ for N_2) ($1 \leq q \leq m$) all are updated to 1 if a solution in the block $B_1[i][j](s)$ (or $B_2[i][j](s)$ for N_2) is chosen as the current solution, and keeps unchanged otherwise. Two illustrative examples for the update of M_1 and M_2 are given in Fig. 3, where each blue entry corresponds to the neighborhood block being examined, those entries in lilac are those that need to be updated if a neighbor solution is chosen to replace the current solution from the block being examined during the neighborhood search.

As we observe in Fig. 3, if a neighbor solution is chosen to replace the current solution during the neighborhood search, only those blocks corresponding to the lilac entries are impacted in terms of the objective value. Thus, the algorithm only needs to examine those blocks with a state value of 1. In this way, the neighborhood search process can be accelerated significantly (see the experimental study on this issue presented in Section 4.1).

2.4.3 Neighborhood Decomposition based Variable Neighborhood Descent

The neighborhood decomposition based variable neighborhood descent procedure (NDVND) (Algorithm 2) relies on the general variable neighborhood descent (VND) method [15]. NDVND employs both neighborhoods N_1 and N_2 to explore candidate solutions. Stating from N_1, the procedure examines N_1 and N_2 in a token-ring way until no improving solution exists in $N_1(s)$ and $N_2(s)$ with respect to the current solution s.

Algorithm 2: Neighborhood decomposition based variable neighborhood descent (NDVND) procedure

```
Function NDVND(s)
Input: Input solution s_0
Output: The local optimum solution s
Initialize state matrices M_1 and M_2                    /* Section 2.4.2 */
Improve ← true
while Improve=true do
    Improve ← false
    /* Examine neighborhood N_1 block by block                          */
    for i ← 1 to m do
        for j ← 1 to m do
            if M_1[i][j] = 1 then
                M_1[i][j] ← 0,flag ← false
                for each s' ∈ B_1[i][j](s) do
                    if f(s') > f(s) then
                        s ← s'
                        Improve ← true, flag ← true
                    end
                end
                if flag = true then
                    Update M_1 and M_2                   /* Section 2.4.2 */
                end
            end
        end
    end
    /* Examine neighborhood N_2 block by block                          */
    for i ← 1 to m do
        for j ← i + 1 to m do
            if M_2[i][j] = 1 then
                M_2[i][j] ← 0, M_2[j][i] ← 0, flag ← false
                for each s' ∈ B_2[i][j](s) do
                    if f(s') > f(s) then
                        s ← s'
                        Improve ← true, flag ← true
                    end
                end
                if flag = true then
                    Update M_1 and M_2                   /* Section 2.4.2 */
                end
            end
        end
    end
end
return s
```

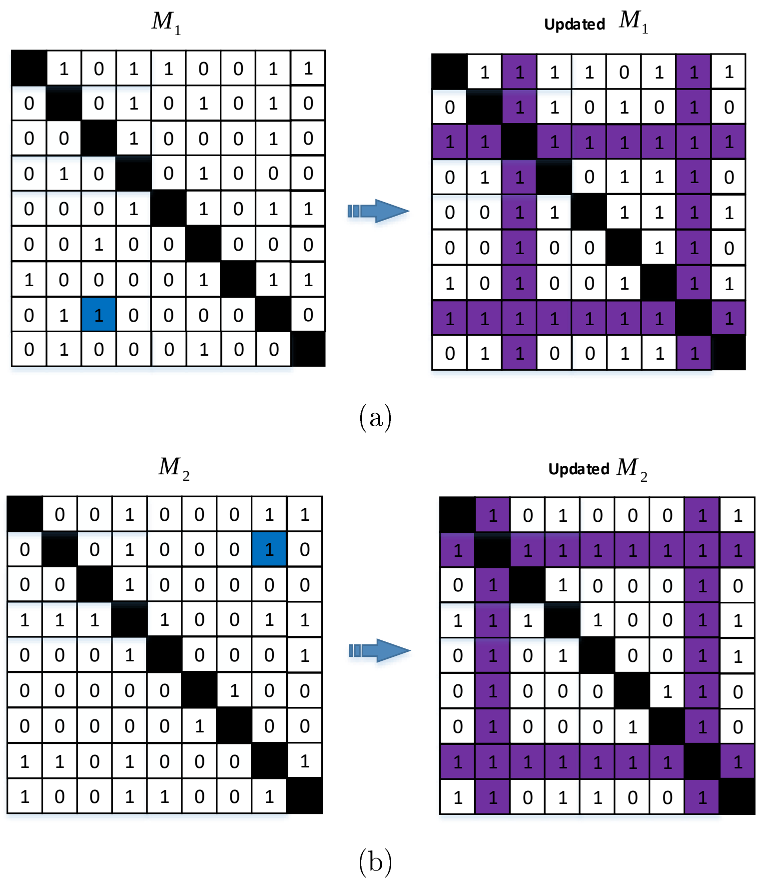


Fig. 3. Illustrative example for the update of the matrices M_1 and M_2, where an blue entry corresponds to the neighborhood block being examined, and the lilac entries are needed to be updated if a neighbor solution in the neighborhood block being examined is chosen to replace the current solution during the neighborhood search.

Specifically, for a given neighborhood N_i $(i = 1, 2)$, it is examined block-by-block, and the current solution s is immediately updated each time an improving neighbor solution is found. Then, the state matrices M_1 and M_2 are updated if at least one improving neighbor solution is found after the associated neighborhood block is completely examined.

2.4.4 Neighborhood Decomposition based Tabu Search

The neighborhood decomposition based tabu search (NDTS) procedure of the NDHA algorithm relies on the tabu search metaheuristic [12] and employs a reduced swap neighborhood N_{NDTS} obtained from the swap neighborhood N_2. Formally, this reduced swap neighborhood is given by $N_{NDTS}(s) = \{s \in B_2[i][j](s) : M_2[i][j] = 1 \vee rand(0,1) < \mu, i < j\}$, where $rand(0, 1)$ is a random number in the interval $(0, 1)$ and μ is a parameter which is set to 0.05 in this study (see analysis of μ in Section 4.3). In other words, the reduced neighborhood includes always the promising neighborhood block $B_2[i][j](s)$ (indicated

Algorithm 3: Neighborhood decomposition based tabu search (NDTS) procedure

```
Function NDTS(s)
Input: Input solution s_0, depth of tabu search α, and parameter μ
Output: The best solution s^b found
s ← s_0                          /* s denotes the current solution */
s^b ← s          /* s^b denotes the best solution found by the tabu
  search */
NoImprove ← 0
Initialize tabu list T and state matrix M_2          /* Section 2.4.2 */
while NoImprove < α do
    /* Examine neighborhood N_NDTS(s) block by block            */
    f_max ← −∞
    for i ← 1 to m do
        for j ← i + 1 to m do
            if (M_2[i][j] = 1) ∨ (rand(0,1) < μ) then
                M_2[i][j] ← 0, M_2[j][i] ← 0
                for each s' ∈ B_2[i][j](s) do
                    if ((f(s') > f_max)∧(s' is not forbidden))∨(f(s') > f(s^b))
                     then
                        f_max ← f(s')
                        I ← i, J ← j
                        s_nb ← s'  /* s_nb denotes the best solution not
                         forbidden by T in N_NDTS(s) */
                    end
                end
            end
        end
    end
    /* Perform the iterations                                    */
    s ← s_nb
    Update tabu list T
    Update state matrix M_2 using I and J          /* Section 2.4.2 */
    if f(s) > f(s^b)) then
        s^b ← s
        NoImprove ← 0
    end
    else
        NoImprove ← NoImprove + 1
    end
end
return s^b
```

with $M_2[i][j] = 1$). To avoid a too restricted (small) neighborhood at each iteration, the neighborhood block $B_2[i][j](s)$ of $N_2(s)$ is always contained in the neighborhood $N_{NDTS}(s)$ with a probability of μ independent of its state ($M_2[i][j] = 1$ or 0).

The NDTS procedure described in Algorithm 3 starts from the initialization of the tabu list T (a $N \times m$ array) and the state matrix M_2 (line 5), then performs a number of iterations until the best solution s^b can not be improved during α consecutive iterations (lines 6–34), where α is a parameter called the depth of tabu search.

At each iteration, the NDTS procedure examines the neighborhood $N_{NDTS}(s)$ in a block-by-block way and chooses a best neighbor solution (denoted by $s_{nb} = s \oplus Swap(v, u)$) that is not forbidden by the tabu list T to replace the current solution s. Then the tabu list T is accordingly updated (line 26), i.e., the corresponding vertices v and u are recorded into T and forbidden to move back to their previous groups for the next tl iterations, where $tl = 15 + rand(5)$ is the tabu tenure with $rand(5)$ being a random integer between 0 and 4. Moreover, the aspiration criterion is applied, i.e., a neighbor solution s_{nb} always replaces the current solution if the quality of s_{nb} is better than the best solution found so far (s^b) (line 14). After that, the state matrix M_2 is accordingly updated (line 26).

2.5 *Perturbation Operator*

Algorithm 4: Perturbation Operator

Function $Perturbation(s_0, \eta)$
 Input: Input solution s_0, strength of perturbation λ
 Output: The perturbed solution s
$s \leftarrow s_0$
for $l \leftarrow 1$ ***to*** λ **do**
 Randomly select two vertices v and u locating at different groups in s
 $s \leftarrow s \oplus Swap(v, u)$ `/* Swap the groups of vertices` v `and` u `*/`
end
return s

To diversify the search process and jump out of local optimum traps, the NDHA algorithm employs a perturbation operator to modify the solutions returned by the local search methods. Specifically, starting from the input solution s_0, the perturbation operator performs a number λ of random swap moves to generate a new solution, where each swap move exchanges the groups of two random vertices located in two distinct groups. As indicated in Section 2.1, the strength of perturbation λ is a changing value which is dynamically adjusted by an adaptive technique. The pseudo-code of the perturbation operator is given in Algorithm 4.

2.6 Discussions on the Innovations of the Work

Compared with the existing iterative algorithms in the literature, such as SGVNS [4], ITS [22] and IMS [18], the proposed NDHA algorithm has two following original features. First, unlike the existing methods where only one local search procedure is used, the proposed NDHA algorithm combines in a probabilistic way two complementary local search procedures (i.e., tabu search and variable neighborhood descent) to reinforce its search robustness on the different types of problem instances. Second, both our tabu search and variable neighborhood descent rely on the innovative neighborhood decomposition strategy to speed up the neighborhood evaluation process, which significantly improves the computational and search efficiency of the algorithm (as shown in Section 3).

Actually, the proposed neighborhood decomposition strategy is the most important innovation for this work, and it is able to increase greatly the computational efficiency of neighborhood examination, without missing improving candidate solutions within the given neighborhood. The neighborhood decomposition strategy follows the general idea of the candidate list strategy to reduce the examined neighborhood size while retaining high-quality solutions. Importantly, it provides a practical and highly effective technique to organize the neighbor solutions into neighborhood blocks and to enable the search procedure to focus on promising neighbor solutions without compromising the quality of the search.

Finally, the idea of neighborhood decomposition is very general and can be advantageously adopted in neighborhood search algorithms for other grouping or clustering problems. Thus, the contribution introduced in this work goes beyond the problem considered here and could potentially benefit many heuristic algorithms for difficult combinatorial optimization.

3 Computational Experiments and Assessments

In this section, we assess the NDHA algorithm by performing large experiments on 320 benchmark instances and making comparisons with state-of-the-art algorithms in the literature.

3.1 Benchmark Instances

In our experiments, we test 320 benchmark instances widely used in the literature. These instances belong to five sets whose main characteristics are summarized as follows [1].

- **RanReal set (20 instances)**: This set includes 10 instances with different group sizes (DGS) and 10 instances with equal group sizes (EGS). For these instances, the number N of elements equals 480 or 960, the number m of groups equals 20 or 24, L_g and U_g vary between 10 and 50, and the distances d_{ij} $(i < j)$ are a real number generated randomly in the interval $(0, 100)$. For the EGS instances, both L_g and U_g are equal to $\lceil N/m \rceil$ for any group g. These instances were tested in [4,11,18,22,23].
- **RanInt set (20 instances)**: Similar to RanReal set, this set contains 10 DGS instances and 10 EGS instances. The main characteristics of these instances are the same as RanReal instances, while the distances d_{ij} $(i < j)$ between elements are an integer generated randomly between 0 and 100. These instances were tested in [4,11,18,22,23].
- **Geo set (20 instances)**: Similar to RanReal and RanInt sets, this set contains 10 EGS instances and 10 DGS instances whose main characteristics are the same as RanReal and RanInt instances, while the distances d_{ij} $(i < j)$ between elements are Euclidean distances between pairs of points with random coordinates from $[0, 10]$, and the number of coordinates of points varies from 2 to 21. These instances were tested in [4,11,18,22,23].
- **MDG-a set (220 instances)**: This set is composed of 11 subsets, including 6 subsets of DGS instances and 5 subsets of EGS instances, and each subset consists of 20 instances that were generated from 20 edge-weighted complete graphs with $N = 2000$, where the edge weights d_{ij} $(i < j)$ are an integer generated randomly between 0 and 10. The main characteristics of these 11 subsets are summarized in Table 1. These instances were tested in [4,18,23].

Table 1
Main characteristics of the instances in the set MDG-a, where '#' indicates the number of instances in the corresponding subset.

		DGS			**EGS**	
N	m	L_g	U_g	#	$L_g = U_g$	#
2000	50	32	48	20	–	–
2000	10	173	227	20	200	20
2000	25	51	109	20	80	20
2000	50	26	54	20	40	20
2000	100	13	27	20	20	20
2000	200	6	14	20	10	20

- **MDG-c set (40 instances)**: This set is composed of 20 DGS instances and 20 EGS instances with $n = 3000$ and $m = 50$, where L_g and U_g are

[1] The benchmark instances as well as the source code of our algorithm will be available at `http://www.info.univ-angers.fr/pub/hao/NDHA.html`

respectively set to $\lfloor 0.8N/m \rfloor$ and $\lceil 1.2N/m \rceil$ for the DGS instances, and $\lceil N/m \rceil$ for the EGS instances. The distances d_{ij} $(i < j)$ between elements are an integer generated randomly between 0 and 1000. These instances are the largest instances used in this study and were first used in [18].

3.2 Parameter Setting and Experimental Protocol

Table 2
Setting of important parameters

Parameters	Section	Description	Values
δ	2.1	parameter used in acceptance criterion	0.01
Q	2.1	probability of applying NDTS and NDVND	0.1
k_{min}	2.1	minimum perturbation strength	$0.2 \times N/m$
k_{max}	2.1	maximum perturbation strength	$2 \times N/m$
k_{step}	2.1	incremental value of perturbation strength	$0.2 \times N/m$
α	2.4.4	depth of tabu search	500
μ	2.4.4	parameter used in the neighborhood N_{NDTS}	0.05

The NDHA algorithm adopts several parameters whose descriptions and settings are listed in Table 2. The value of each parameter was fixed independently according to a preliminary experiment performed on a selection of instances of different characteristics. Typically, we tested a number of possible values from a given range to retain the value leading to the best average result. We observe that among the parameters, Q which controls the probability of applying the NDVND or NDTS optimization procedure is one key parameter, for which we provide a detailed analysis in Section 4.2. Notice that the parameter values shown in Table 2 can be considered to be the default parameter setting of the NDHA algorithm, which were used consistently to perform all the experiments reported in this work unless stated otherwise.

According to the computational results reported in [4,18,22,24], the algorithms ITS [22], SGVNS [4], IMS [18], and NSGGA [24] (only for instances with equal group sizes) outperform significantly other algorithms in the literature and can be regarded as the state-of-the-art algorithms for MDGP. Hence, in this work, we use these algorithms as the reference algorithms. Among these reference algorithms, the source code of ITS is available at `http://www.proin.ktu.lt/~gintaras/mdgp.html`, the source code of IMS is available at `http://www.info.univ-angers.fr/pub/hao/mdgp.html`, and the executable code of SGVNS was kindly provided by the authors of [4]. The proposed NDHA algorithm as well as IMS and ITS were written in C++ and compiled using the g++ compiler with the -O3 option.

In addition, all the computational experiments were carried out on the same computing platform with an Intel E5-2670 processor (2.5 GHz and 2G RAM), running the Linux operating system. Following [18], the stopping condition of all the algorithms is a cutoff time limit t_{max} set to 120, 600, 1200, and 3000 seconds for instances $n = 480$, $n = 960$, $n = 2000$ and $n = 3000$, respectively. Finally, to assess the average performance of the algorithms, the NDHA algorithm and the reference algorithms (i.e., ITS, SGVNS, IMS) were performed 20 times with different random seeds for each run.

3.3 Computational Results and Comparison on the Small Instances

The first experiment is devoted to an assessment of the NDHA algorithm on the 60 small instances with $N \leq 960$, from RanInt, RanReal and Geo sets. The experimental results of the reference algorithms (ITS, SGVNS, IMS) as well as our NDHA algorithm are summarized in Table 3 (with different group sizes) and Table 4 (with equal group sizes). Columns 2–5 of each table give the best objective value (f_{best}) over 20 runs respectively for the compared algorithms, and columns 6–9 show the average objective value (f_{avg}). The row 'Avg.' indicates the average result for each column, and the row '#best' indicates the number of instances for which the corresponding algorithm obtained the best result among the compared algorithms in terms of f_{best} and f_{avg}. The best f_{best} and f_{avg} values among the compared algorithms are indicated in bold for each instance. In addition, to check the statistical difference between NDHA and each reference algorithm in terms of f_{best} or f_{avg}, the p-values from the Wilcoxon signed-rank tests are reported in the last row of the tables, and a p-value smaller than 0.05 means that there exists a significant difference between the NDHA algorithm and the corresponding reference algorithm.

Table 3 shows that for the instances with $N \leq 960$ and different group sizes, the NDHA algorithm is very competitive compared to the reference algorithms. In terms of f_{best}, the ITS, SGVNS, IMS and NDHA algorithms obtained respectively the best result for 3, 0, 15 and 12 out of 30 instances. In terms of f_{avg}, these four algorithms obtained the best result for 1, 0, 15 and 14 instances. Moreover, the small *p-values* (≤ 0.05) mean that the NDHA algorithm outperforms significantly the ITS and SGVNS algorithms both in terms of f_{best} and f_{avg}. When comparing with IMS and NDHA, we observe that they perform similarly (*p-values* ≥ 0.05) in terms of f_{best} and f_{avg}.

Table 4 indicates that for the instances with $N \leq 960$ and equal group sizes, the NDHA algorithm performs well compared to the reference algorithms. In terms of f_{best}, the ITS, SGVNS, IMS and NDHA algorithms obtained respectively the best result for 11, 0, 8 and 11 out of 30 instances. The large *p-values* (7.19E-2 and 4.53E-1) imply that there does not exist a significant difference between NDHA and ITS (or IMS). However, the small *p-value* (2.76E-3) means that NDHA performs significantly better than the SGVNS algorithm. In terms of f_{avg}, the *p-values* show that the NDHA algorithm outperforms significantly the ITS and SGVNS algorithms and performs similarly compared to the IMS algorithm.

Table 3
Comparison of the proposed NDHA algorithm with three state-of-the-art algorithms in the literature on the small instances with different group sizes. The best results between the compared algorithms in terms of f_{best} and f_{avg} are indicated in bold.

	f_{best}				f_{avg}			
Instance	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
Geo_n480_ds_01	580908.19	579594.38	582379.10	**582587.38**	580637.81	577253.71	580531.36	**582339.19**
Geo_n480_ds_02	1089063.79	1088385.58	1089873.95	**1090135.64**	1088309.08	1082940.35	1088725.10	**1089903.18**
Geo_n480_ds_03	662675.55	661554.64	664243.52	**664483.97**	662014.57	658540.72	662886.91	**664164.89**
Geo_n480_ds_04	**836700.53**	835426.31	836324.69	836547.59	**836561.41**	831787.05	835091.09	836410.06
Geo_n480_ds_05	**988491.52**	987106.70	988261.34	988444.21	986781.87	979093.78	985494.57	**988173.67**
RanInt_n480_ds_01	389548.00	390260.00	**390326.00**	390103.00	387744.10	389015.10	389517.40	**389549.00**
RanInt_n480_ds_02	387757.00	388404.00	**389019.00**	388644.00	386570.25	387487.00	**388228.00**	387832.30
RanInt_n480_ds_03	387751.00	387615.00	**388756.00**	388182.00	386103.15	386771.10	**387615.45**	387222.35
RanInt_n480_ds_04	391337.00	391219.00	**392124.00**	392091.00	389776.20	390362.45	**391382.75**	391323.80
RanInt_n480_ds_05	388331.00	389330.00	**389448.00**	389300.00	387119.25	387972.75	**388655.05**	388334.10
RanReal_n480_ds_01	387803.44	387929.42	388597.02	**388841.84**	385976.58	386955.60	387786.73	**387900.09**
RanReal_n480_ds_02	386295.34	386497.74	**387003.53**	386751.97	384832.11	385730.85	**386147.90**	385562.71
RanReal_n480_ds_03	387474.69	387900.83	**389154.79**	387746.68	386066.36	386911.54	**387760.16**	387098.28
RanReal_n480_ds_04	390225.82	390244.43	390547.26	**390587.80**	388485.81	389185.26	**390155.96**	390052.19
RanReal_n480_ds_05	387831.00	387688.36	**388256.59**	388233.84	385946.71	386532.01	**387557.27**	387062.41
Geo_n960_ds_01	3361972.63	3352319.16	3364802.56	**3365362.95**	3348637.43	3342837.38	3356273.08	**3363776.03**
Geo_n960_ds_02	1719892.54	1720029.46	1722401.13	**1723727.15**	1718034.17	1713409.47	1719324.79	**1722784.10**
Geo_n960_ds_03	3347803.20	3346546.58	3350871.47	**3351599.56**	3347081.90	3337399.48	3345799.89	**3350888.19**
Geo_n960_ds_04	3615110.26	3622509.71	3622341.03	**3623707.59**	3603672.09	3608470.86	3617908.67	**3622936.94**
Geo_n960_ds_05	**2342675.15**	2337386.27	2342091.85	2342515.47	2342347.12	2329478.25	2337198.18	**2341996.35**
RanInt_n960_ds_01	1240283.00	1239528.00	1243072.00	**1243208.00**	1233592.20	1237318.10	**1242268.05**	1241229.25
RanInt_n960_ds_02	1237216.00	1237063.00	**1241131.00**	1240486.00	1231173.05	1235400.05	**1239855.70**	1238684.85
RanInt_n960_ds_03	1236892.00	1237683.00	1241296.00	**1241878.00**	1231942.40	1235185.25	1239497.15	**1239675.50**
RanInt_n960_ds_04	1237828.00	1238868.00	**1241649.00**	1241411.00	1233270.40	1236203.80	**1240409.45**	1239776.55
RanInt_n960_ds_05	1237852.00	1237764.00	**1241342.00**	1241024.00	1232948.10	1236090.90	1239319.35	**1239658.85**
RanReal_n960_ds_01	1235611.95	1235871.43	**1240525.13**	1240269.22	1231084.73	1234109.07	**1239157.73**	1238140.86
RanReal_n960_ds_02	1234529.41	1236135.97	**1240159.33**	1239398.55	1230281.42	1234199.50	**1238546.22**	1237526.20
RanReal_n960_ds_03	1234256.69	1234244.88	**1238956.75**	1238053.93	1229824.54	1233000.51	**1237019.78**	1236872.79
RanReal_n960_ds_04	1235440.27	1236451.51	**1239641.15**	1239497.50	1229716.17	1234727.43	**1238624.95**	1238095.56
RanReal_n960_ds_05	1232976.13	1233629.07	1236955.98	**1238382.45**	1229769.51	1231419.21	1235698.07	**1236018.17**
Avg.	1159751.10	1159506.21	1162051.71	**1162106.74**	1156543.35	1156192.95	1160147.89	**1161032.95**
#best	3	0	**15**	12	1	0	**15**	14
p-value	3.18E-6	3.18E-6	8.45E-1		2.35E-6	1.92E-6	3.09E-1	

3.4 Computational Results and Comparison on the Large Scale Instances

The second experiment aims to assess the NDHA algorithm on the 260 large scale instances belonging to 11 subsets of the set MDG-a and 2 subsets of the set MDG-c, where each subset has 20 instances with $N = 2000$ or $N = 3000$. For the experiment, the NDHA algorithm and the reference algorithms (ITS, SDVNS, and IMS) were respectively run 20 times on each instance, and the detailed results for each subset are reported in Tables A.1–A.13 of the appendix, where the same information as in Tables 3 and 4 is provided.

A summary of these detailed experimental results is provided in Table 5, where each row represents one subset. Columns 1–5 of the table give the main characteristics of the instances in the corresponding subset, columns 6–9 indicate the average results (i.e., the *Avg.* value in Tables A.1–A.13) in terms of f_{best} respectively for each subset and each algorithm, and columns 10–13 shows the average results in terms of f_{avg} for each subset and each algorithm. The last row of the table indicates the number of instances for which the associated algorithm obtained the best result in terms of f_{best} and f_{avg} among the compared algorithms.

Table 4
Comparison of the proposed NDHA algorithm with three state-of-the-art algorithms in the literature on the small instances with equal group sizes. The best results between the compared algorithms in terms of f_{best} and f_{avg} are indicated in bold.

	f_{best}				f_{avg}			
Instance	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
Geo_n480_ss_01	**552206.89**	552040.71	552073.45	552194.91	552165.47	552020.33	552045.83	**552167.92**
Geo_n480_ss_02	**1047462.35**	1047245.92	1047228.47	1047433.23	**1047405.08**	1047155.51	1047182.54	1047322.44
Geo_n480_ss_03	**633855.88**	633574.99	633626.24	633740.92	**633746.62**	633544.22	633590.50	633713.2514
Geo_n480_ss_04	**789891.16**	789621.27	789657.63	789767.10	**789791.95**	789544.51	789610.72	789731.7703
Geo_n480_ss_05	**945974.03**	945667.90	945782.66	945865.98	**945898.63**	945643.47	945697.53	945819.7054
RanInt_n480_ss_01	379408.00	379532.00	**380127.00**	377481.62	378288.25	**378944.75**	378874.75	376897.68
RanInt_n480_ss_02	379682.00	379465.00	**379978.00**	376915.63	377891.60	378806.60	**379193.70**	376396.93
RanInt_n480_ss_03	378616.00	378677.00	**378691.00**	378170.99	377267.90	**378021.70**	377954.45	377516.86
RanInt_n480_ss_04	378416.00	378582.00	**378761.00**	377288.47	377041.30	378082.75	**378117.80**	376712.10
RanInt_n480_ss_05	**379627.00**	379533.00	379420.00	378693.73	377627.10	378697.00	**378697.55**	377348.96
RanReal_n480_ss_01	377744.50	377553.91	377737.96	**379167.00**	376224.21	376996.03	377233.60	**378564.90**
RanReal_n480_ss_02	377306.86	377342.67	377203.56	**379708.00**	375845.53	376843.49	376598.81	**379053.90**
RanReal_n480_ss_03	378407.16	378516.70	**379278.21**	378323.00	376942.00	**377990.35**	377908.31	377751.95
RanReal_n480_ss_04	377512.67	377483.34	377447.86	**378915.00**	376176.89	376817.41	377041.65	**377989.75**
RanReal_n480_ss_05	377708.04	378113.82	378227.50	**378845.00**	376305.51	377363.53	377559.44	**378352.70**
Geo_n960_ss_01	**3254625.15**	3253979.37	3254027.55	3254394.48	**3254341.67**	3253886.64	3253941.37	3254256.201
Geo_n960_ss_02	**1663654.73**	1663474.52	1663486.06	1663618.41	**1663602.47**	1663443.67	1663432.13	1663584.936
Geo_n960_ss_03	**3251862.13**	3251193.24	3251319.45	3251595.05	**3251574.46**	3251122.52	3251201.99	3251496.215
Geo_n960_ss_04	**3514547.37**	3513915.12	3513974.50	3514238.95	**3514260.32**	3513828.00	3513830.30	3514168.624
Geo_n960_ss_05	**2264972.55**	2264438.39	2264501.25	2264774.65	2264721.05	2264405.61	2264462.56	**2264725.134**
RanInt_n960_ss_01	1218147.00	1217689.00	**1220591.00**	1220052.00	1213208.80	1216154.50	**1219093.15**	1218752.50
RanInt_n960_ss_02	1215854.00	1216546.00	1219844.00	**1221550.00**	1211598.85	1215158.85	**1218455.75**	1218307.50
RanInt_n960_ss_03	1217134.00	1217434.00	1220172.00	**1220874.00**	1212712.85	1216107.65	**1219097.80**	1219068.20
RanInt_n960_ss_04	1216532.00	1217806.00	1220605.00	**1221490.00**	1212683.70	1215891.40	**1219315.80**	1218413.20
RanInt_n960_ss_05	1215621.00	1216975.00	1220574.00	**1220771.00**	1212488.15	1215436.75	1218694.05	**1218718.75**
RanReal_n960_ss_01	1214107.46	1214382.40	**1218358.90**	1216886.68	1210716.76	1212547.12	**1216271.17**	1215373.16
RanReal_n960_ss_02	1214784.09	1215430.98	**1218858.40**	1218587.92	1211066.99	1213585.65	**1217280.82**	1216566.55
RanReal_n960_ss_03	1214095.83	1215095.58	1218106.72	**1218341.93**	1209634.07	1213582.73	**1216601.52**	1215722.71
RanReal_n960_ss_04	1214677.73	1215275.81	1219153.05	**1219556.92**	1211544.50	1214126.05	**1217344.30**	1217283.14
RanReal_n960_ss_05	1212727.73	1213650.67	1216750.56	**1216923.18**	1208607.96	1211717.26	1215229.52	**1215287.53**
Avg.	1128572.04	1128674.54	1129852.10	**1129872.19**	1126712.69	1127915.53	**1129051.98**	1128902.17
#best	**11**	0	8	**11**	8	3	**11**	8
p-value	7.19E-2	2.76E-3	4.53E-1		1.04E-2	3.16E-3	5.86E-1	

Table 5
Summary comparison of the NDHA algorithm with three state-of-the-art algorithms on 260 large instances with $n = 2000$ (220 instances) and $n = 3000$ (40 instances). The best results between the compared algorithms in terms of f_{best} and f_{avg} are indicated in bold.

Instance sets					f_{best}				f_{avg}			
Set	m	L_g	U_g	*Type*	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
MDG-a	10	173	227	DGS	1134281.45	1132567.10	**1135887.90**	1135492.75	1132625.40	1131706.31	**1135362.83**	1134854.76
MDG-a	10	200	200	EGS	1115486.45	1113931.55	**1117171.70**	1116895.05	1114055.35	1113142.01	**1116717.32**	1116269.40
MDG-a	25	51	109	DGS	539452.05	539586.85	541217.2	**541406.85**	538228.13	539098.47	540803.28	**540936.41**
MDG-a	25	80	80	EGS	486000.1	486159.75	487575.4	**487923.55**	484766.67	485663.39	487242.80	**487538.23**
MDG-a	50	26	54	DGS	291005.4	291821.7	292609.15	**293188.65**	289955.77	291441.89	292308.39	**292847.22**
MDG-a	50	32	48	DGS	272851.65	273522.70	274397.30	**274898.50**	271787.65	273221.62	274039.12	**274506.32**
MDG-a	50	40	40	EGS	263719.45	264561.50	265437.95	**265943.15**	262679.20	264262.92	265119.75	**265560.06**
MDG-a	100	13	27	DGS	158745.10	159297.10	159811.60	**160813.85**	157897.57	159015.00	159557.94	**160549.57**
MDG-a	100	20	20	EGS	143917.30	144602.45	144834.50	**145476.90**	143088.89	144331.32	144586.34	**145161.32**
MDG-a	200	6	14	DGS	88303.85	88512.20	88829.60	**89598.45**	87733.85	88326.22	88599.92	**89419.56**
MDG-a	200	10	10	EGS	76972.90	76841.70	77221.95	**78276.90**	76467.98	76689.70	77050.59	**78116.26**
MDG-c	50	48	72	DGS	57907172.80	58047799.55	58233404.15	**58324705.90**	57732451.10	57999123.67	58186201.46	**58274147.92**
MDG-c	50	60	60	EGS	55914269.20	56016286.80	56269513.80	**56366981.20**	55762202.26	55970537.94	56228021.37	**56320965.85**
#best					0	0	38	**222**	0	0	40	**220**

Table 5 shows that the NDHA algorithm performs very well and outperforms significantly the reference algorithms. The NDHA algorithm obtained the best results for 222 and 220 out of 260 instances in terms of f_{best} and f_{avg}, respectively. Compared to the ITS and SGVNS algorithms, our algorithm obtained a better result for each instance both in terms of f_{best} and f_{avg}. When com-

Table 6
Comparison of the NDHA algorithm with the NSGGA algorithm on 20 EGS instances with $N = 2000$ and $m = 200$. The best results between the compared algorithms in terms of f_{best} and f_{avg} are indicated in bold.

Instances				f_{best}		f_{avg}	
Graph	m	L_g	U_g	NSGGA	NDHA	NSGGA	NDHA
MDG-a_21	200	10	10	77610	**78193**	77299.80	**78101.00**
MDG-a_22	200	10	10	77671	**78423**	77290.50	**78098.35**
MDG-a_23	200	10	10	77567	**78253**	77271.30	**78111.00**
MDG-a_24	200	10	10	77401	**78300**	77213.35	**78075.35**
MDG-a_25	200	10	10	77536	**78266**	77317.30	**78143.55**
MDG-a_26	200	10	10	77442	**78324**	77263.25	**78107.90**
MDG-a_27	200	10	10	77510	**78220**	77241.30	**78085.00**
MDG-a_28	200	10	10	77670	**78208**	77290.85	**78107.75**
MDG-a_29	200	10	10	77442	**78271**	77242.60	**78104.90**
MDG-a_30	200	10	10	77575	**78187**	77272.60	**78092.05**
MDG-a_31	200	10	10	77557	**78380**	77323.55	**78255.45**
MDG-a_32	200	10	10	77470	**78252**	77245.05	**78117.95**
MDG-a_33	200	10	10	77480	**78234**	77271.85	**78085.05**
MDG-a_34	200	10	10	77538	**78193**	77327.65	**78082.05**
MDG-a_35	200	10	10	77684	**78332**	77334.60	**78094.25**
MDG-a_36	200	10	10	77519	**78348**	77273.10	**78158.55**
MDG-a_37	200	10	10	77630	**78335**	77373.85	**78126.90**
MDG-a_38	200	10	10	77493	**78189**	77269.00	**78100.00**
MDG-a_39	200	10	10	77461	**78290**	77228.35	**78122.50**
MDG-a_40	200	10	10	77544	**78340**	77321.50	**78155.55**
Avg.				77540.00	**78276.90**	77283.57	**78116.26**
#Better				0	**20**	0	**20**
p-value				8.86E-5		8.86E-5	

paring with the IMS algorithm, the NDHA algorithm yielded a better result both in terms of f_{best} and f_{avg} on most instances except the instances with $m = 10$, which means that the neighborhood decomposition strategy used in the NDHA algorithm plays an important role in enhancing the performance of the algorithm for the instances with a large number of groups. This experiment indicates that the proposed NDHA algorithm is highly competitive compared to the three reference algorithms especially for the large instances with a large number of elements and groups.

In addition, to make a comparison between the NDHA algorithm and the recent hybrid genetic algorithm NSGGA which is designed for MDGP with equal group sizes [24], we provide the results of the NSGGA and NDHA algorithms in Table 6 for 20 large EGS instances with $N = 2000$ and $m = 200$, where the same statistical information is given as in the previous tables. Since the code of the NSGGA algorithm is not available to us, this comparison is based on the results reported in [24], based 20 runs per instance on an Intel Core i5 computer with 4G RAM under the same stopping condition as that used by our NDHA algorithm (specified in Section 3.2). One observes from Table 6 that the NDHA algorithm significantly dominates the NSGGA algorithm in terms of f_{best} and f_{avg}, which is confirmed by small *p-values*.

In summary, the experimental results shown in this section indicate that the NDHA algorithm is very competitive compared with the state-of-the-art algorithms in the literature and performs especially well on the large instances.

4 Analysis and Discussions

In this section, we analyze two essential components of the proposed algorithm, i.e., the neighborhood decomposition strategy employed by the local search methods and the strategy of jointly using two local search methods.

4.1 Importance of the Neighborhood Decomposition Strategy

In order to show the effectiveness of the neighborhood decomposition strategy introduced in this study, we carried out an experiment based on 20 large EGS instances with $n = 2000$, $m = 200$ and $L_g = U_g = 10$. For this study, we created a variant (denoted by NDHA-D) of the NDHA algorithm by disabling the neighborhood decomposition strategy (i.e., setting all entries of state matrices M_1 and M_2 to the value of 1 during the neighborhood search process), while keeping other components unchanged. We ran NDHA-D and NDHA 20 times for each instance according to the experimental protocol of Section 3.2. The experimental results are summarized in Table 7, where columns 1–4 give the name and main characteristics of the instances, columns 5–6 show respectively the best objective value (f_{best}) over 20 runs for the two compared algorithms, columns 7–8 indicate the average objective value f_{avg}, and columns 9–10 present the worst objective value (f_{worst}). In addition, the row 'Avg.' shows the average result for each column, the row '#better' shows the number of instances for which the associated algorithm obtained a better result than the competing algorithm in terms of f_{best}, f_{avg}, and f_{worst}, respectively, and the last row gives the *p-values* from the Wilcoxon signed-rank tests for the compared algorithms. The better results between the compared algorithms are indicated in bold.

One observes from Table 7 that the NDHA algorithm dominates the NDHA-D algorithm for each considered performance indicator. Specifically, the NDHA algorithm obtained a better result than the NDHA-D algorithm for all tested instances in terms of f_{best}, f_{avg}, and f_{worst}. Moreover, the small *p-values* imply the difference between the two compared algorithms is statistically significant. This experiment shows the neighborhood decomposition strategy used in this study plays an important role for the high performance of the NDHA algorithm.

To further show the effect of the neighborhood decomposition strategy on the tabu search procedure that is one main component of the NDHA algorithm, we carried out another experiment to compare the tabu search procedure with neighborhood decomposition (i.e., NDTS) and a tabu search procedure without neighborhood decomposition (denoted by TS). To ease the presentation,

Table 7
Comparison between the NDHA algorithm and its variant NDHA-D on 20 EGS instances with $N = 2000$ and $m = 200$.

				f_{best}		f_{avg}		f_{worst}	
Graph	m	L_g	U_g	NDHA-D	NDHA	NDHA-D	NDHA	NDHA-D	NDHA
MDG-a_21	200	10	10	77801	**78193**	77508.40	**78101.00**	77275	**77990**
MDG-a_22	200	10	10	77715	**78423**	77527.95	**78098.35**	77331	**77983**
MDG-a_23	200	10	10	77681	**78253**	77448.80	**78111.00**	77189	**77940**
MDG-a_24	200	10	10	77897	**78300**	77489.70	**78075.35**	77183	**77755**
MDG-a_25	200	10	10	77870	**78266**	77503.15	**78143.55**	77245	**77937**
MDG-a_26	200	10	10	77765	**78324**	77569.30	**78107.90**	77235	**77928**
MDG-a_27	200	10	10	77661	**78220**	77431.70	**78085.00**	77303	**77914**
MDG-a_28	200	10	10	77700	**78208**	77457.70	**78107.75**	77210	**77996**
MDG-a_29	200	10	10	77734	**78271**	77510.00	**78104.90**	77238	**77947**
MDG-a_30	200	10	10	77766	**78187**	77528.15	**78092.05**	77329	**77967**
MDG-a_31	200	10	10	77765	**78380**	77517.25	**78255.45**	77316	**78144**
MDG-a_32	200	10	10	77702	**78252**	77540.80	**78117.95**	77303	**77960**
MDG-a_33	200	10	10	77789	**78234**	77485.55	**78085.05**	77249	**77905**
MDG-a_34	200	10	10	77725	**78193**	77534.80	**78082.05**	77359	**77956**
MDG-a_35	200	10	10	77638	**78332**	77465.95	**78094.25**	77238	**77978**
MDG-a_36	200	10	10	77685	**78348**	77543.00	**78158.55**	77389	**77890**
MDG-a_37	200	10	10	77805	**78335**	77533.65	**78126.90**	77274	**77964**
MDG-a_38	200	10	10	77698	**78189**	77521.10	**78100.00**	77238	**77947**
MDG-a_39	200	10	10	77785	**78290**	77457.75	**78122.50**	77220	**77969**
MDG-a_40	200	10	10	77766	**78340**	77490.45	**78155.55**	77365	**78019**
Avg.				77747.40	**78276.90**	77503.26	**78116.26**	77274.45	**77954.45**
#Better				0	**20**	0	**20**	0	**20**
p-value				8.86E-5		8.86E-5		8.86E-5	

we illustrate this experiment with the results obtained by NDTS and TS on two large instances (i.e., MDG-a_21 with N=2000, $m = 200$ and $L_g = U_g = 10$ for any g, and MDG-a_24 with $N = 2000$, $m = 50$, $L_g = 26$, and $U_g = 54$ for any g). For the experiment, NDTS and TS were performed one time per instance, starting from the same initial solution. The experimental results are shown in Fig. 4, where the subfigures (a) and (c) show the running times of NDTS and TS as a function of the number of iterations, and the subfigures (b) and (d) show their best objective values $f(s)$ found as a function of running time.

The subfigures (a) and (c) of Fig. 4 indicate that NDTS (with neighborhood decomposition) requires much less time than TS (without neighborhood decomposition) to perform the same number of iterations, which means that the neighborhood decomposition technique is able to speed up the neighborhood search process notably. On the other hand, one observes from the subfigures (b) and (d) that NDTS yielded much better results in the objective value $f(s)$ than TS within the same computation time, implying the neighborhood decomposition technique is able to enhance significantly the performance of the tabu search procedure.

4.2 Impact of Hybridizing Two Local Search Methods

To enhance its robustness for solving instances with very different characteristics, the NDHA algorithm hybridizes two local search procedures (i.e., NDVND and DNTS) in a probabilistic way, where the probability is con-

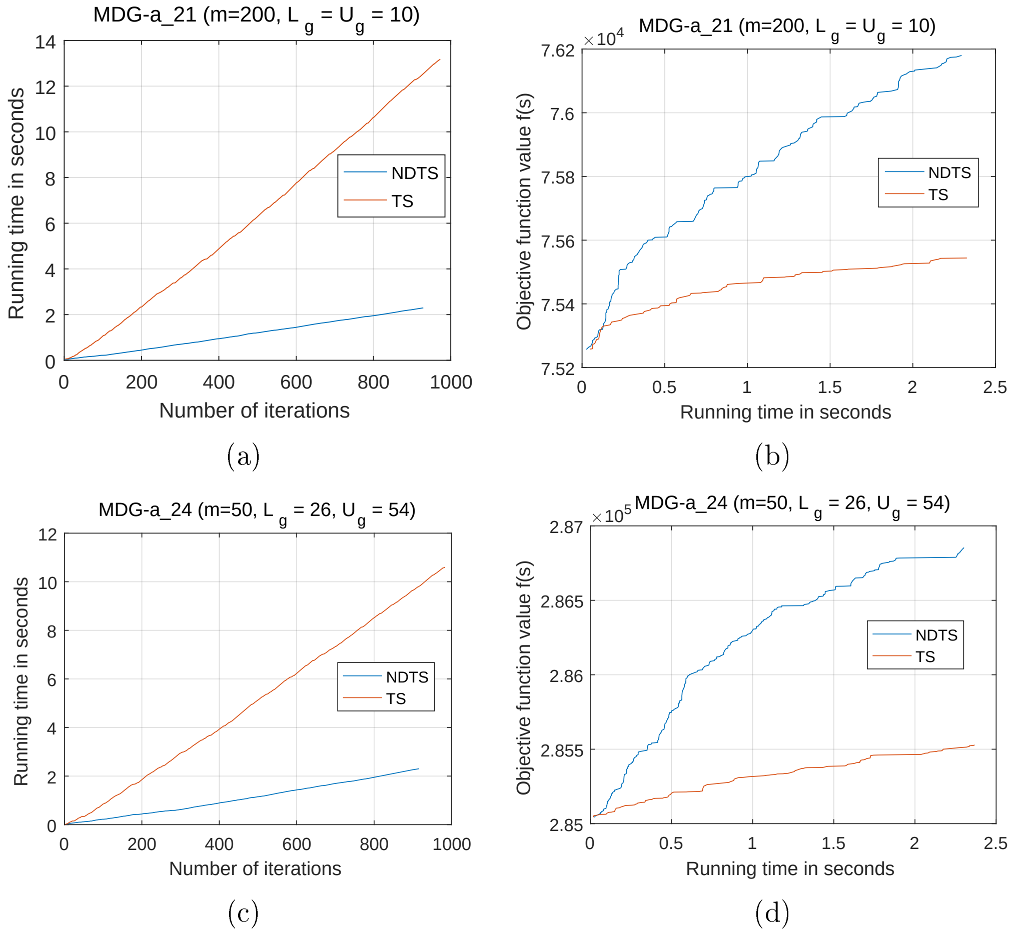
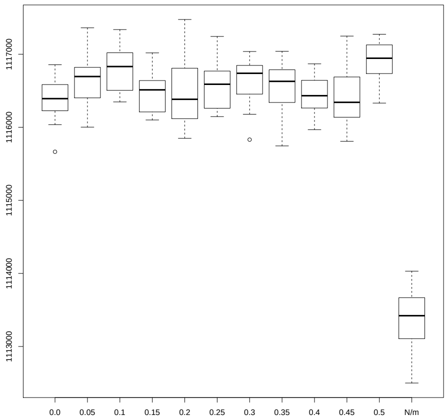


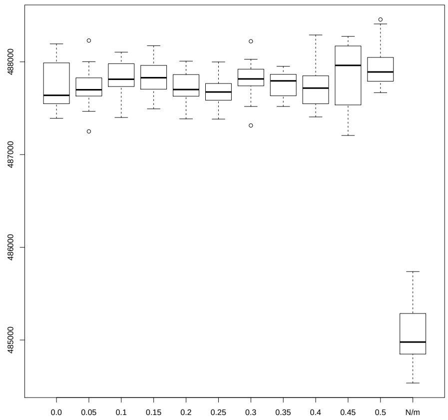
Fig. 4. Comparison between NDTS and TS in terms of the objective function value $f(s)$ and the running time.

trolled by the parameter Q. To show the rationality of this hybridization and choose an appropriate value for Q, we carried out an additional experiment based on five representative EGS instances with very different numbers (m) of groups. For each considered instance and each Q value in the range $\{0.0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, N/m\}$, the NDHA algorithm was performed 20 times using the experimental protocol in Section 3.2. The experimental results are summarized in Fig. 5 using the box and whisker plots, where the X-axis indicates the values of Q and the Y-axis gives the objective values.

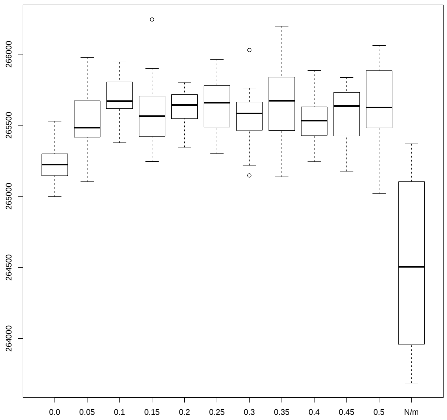
Fig. 5 shows that the performance of the algorithm is sensitive to the setting of Q. Specifically, when the neighborhood decomposition based tabu search (NDTS) procedure was always applied and the neighborhood decomposition based variable neighborhood descent (NDVND) procedure was disabled (with $Q = N/m$), the NDHA algorithm yielded the worst results among the considered Q values for the instances with a small number ($m \leq 50$) of groups. On the other hand, when only the NDVND procedure was applied and the



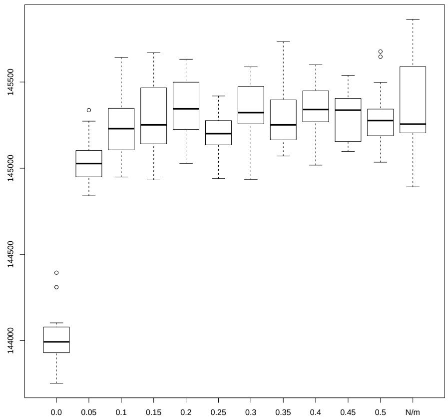
(a) MDG-a_21 with $m = 10$



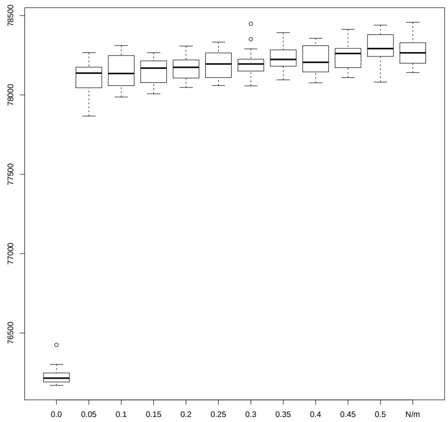
(b) MDG-a_21 with $m = 25$



(c) MDG-a_21 with $m = 50$



(d) MDG-a_21 with $m = 100$



(e) MDG-a_21 with $m = 200$

Fig. 5. Sensitivity analysis for the parameter Q used to control the probability that the NDTS procedure is applied. The X-axis indicates the settings of Q and the Y-axis indicates the objective value.

NDTS procedure was disabled (with $Q = 0$), the NDHA algorithm produced the worst results among the considered Q values for the instances with a large number ($m \geq 100$) of groups. The experiment shows that the NDTS procedure is more suitable for the instances with a large number of groups, while the

NDVND procedure is more suitable for the instances with a small number of groups. This provides the main motivation of combining NDVND and DNTS in a probabilistic way to be able to deal with instances with small and large number of groups. One notices that the setting of $Q = 0.1$ led globally to a good performance for the considered instances, which was used as the default value of Q in this work.

4.3 Sensitivity Analysis of Parameter μ Used in Neighborhood Decomposition based Tabu Search

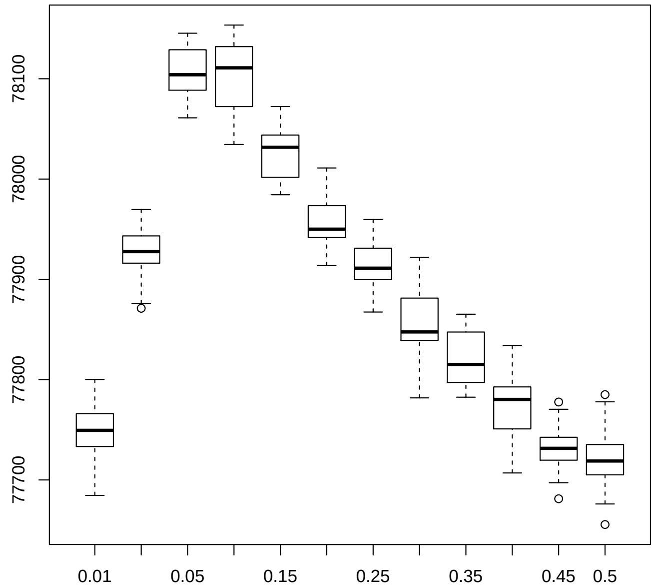


Fig. 6. Sensitivity analysis of parameter μ based on the 20 EGS instances with $N = 2000$ and $m = 200$, where the X-axis indicates the parameter values and the Y-axis shows the average objective values (f_{avg}) obtained over 20 independent runs.

The neighborhood decomposition based tabu search procedure described in Section 2.4.4 employs a key parameter μ to control the size of neighborhood $N_{NDTS}(s)$. In general, a larger μ value leads to a larger neighborhood, and vice verse. To analyze the influence of this parameter on the performance of the algorithm, we conducted an experiment based on the 20 EGS instances with $N = 2000$ and $m = 200$. In this experiment, we ran the NDHA algorithm 20 times for each instance and each μ value in the range of $\{0.01, 0.02, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.4, 0.45, 0.5\}$, and recorded the average objective values (f_{avg}). Fig. 6 shows the results with the popular box and whisker plots, where the X-axis indicates the value of parameter μ and the Y-axis shows the average objective values f_{avg} for the 20 instances.

Fig. 6 indicates that the setting of parameter μ significantly impacts the behavior of the NDHA algorithm. Specifically, For $\mu < 0.05$, the performance of the algorithm gradually increases with the increase of μ, while the performance decreases as the value of μ increases for $\mu \geq 0.15$. Thus, $[0.05, 0.10]$ is a suitable range for μ and 0.05 was adopted as the default value in this work.

5 Conclusions and Future Work

We presented a new heuristic algorithm (NDHA) for solving the maximally diverse grouping problem (MDGP). The proposed algorithm distinguishes itself from existing algorithms by its speeding-up neighborhood decomposition technique and the joint use of two complementary local search procedures (both are based on neighborhood decomposition). An adaptive perturbation strategy is additionally used to escape local optimum traps. The algorithm was assessed on 320 benchmark instances commonly used in the literature. Our computational results show that the algorithm outperforms significantly the state-of-the-art algorithms by reporting improved lower bounds for 220 large benchmark instances (i.e., for more than 68% of the tested instances). Given that MDGP is a general model able to formulate a number of real-world applications, the proposed algorithm provides a valuable tool to better solve the related practical problems. The availability of the source code of our algorithm further facilitates such applications.

Additional analyses indicate that the effectiveness of the algorithm is mainly attributed to two essential ideas, i.e., the neighborhood decomposition strategy which speeds up the neighborhood search process and the stochastic combination strategy of two local search procedures which enhances the robustness of the algorithm for different types of problem instances.

The ideas of neighborhood decomposition and combination of different local search procedures are rather general. It is interesting to verify their usefulness for solving other related grouping or clustering problems, such as the capacitated p-median problem [7], the normalized cut clustering problem [8], and the balanced k-means clustering problem [6]. Moreover, the proposed algorithm can be further reinforced by following two directions. First, the current algorithm does not explicitly deal with the issue of solution symmetry for the case where some or all groups have an equal group size. It is meaningful to investigate dedicated search strategies and techniques exploiting the symmetry property. Second, population-based methods such as memetic computing are known to be general frameworks with the potential of surpassing local search algorithms [14]. Then it is worth designing such hybrid algorithms where the proposed algorithm plays the key role of local optimization for intensification.

Acknowledgments

We are grateful to the reviewers for their valuable comments and suggestions which helped us to improve the paper. This work was partially supported by the National Natural Science Foundation of China (Grant 61703213), the Natural Science Foundation of Jiangsu Province of China (Grant BK20170904), six talent peaks project in Jiangsu Province (Grant RJFW-011), and the Shenzhen Science and Technology Innovation Council (Grant JCYJ20180508162601910) and Shenzhen Institute of Artificial Intelligence and Robotics for Society (2019-INT003).

References

[1] Arani T., Lofti V., 1989, A three phased approach to final exam scheduling. *IIE Transactions*, 21(1), 86–96.

[2] Benlic U., Hao J.K., 2013, Breakout local search for the vertex separator problem. In F. Rossi (Ed.): Proc. of the 23th Intl. Joint Conference on Artificial Intelligence (IJCAI-13), IJCAI/AAAI Press, pages 461-467, Beijing, China

[3] Benlic U., Hao J.K., 2013, Breakout local search for the max-cut problem. Engineering Applications of Artificial Intelligence, 26(3), 1162-1173.

[4] Brimberg J., Mladenović N., Urošević D., 2015, Solving the maximally diverse grouping problem by skewed general variable neighborhodd search. *Information Sciences*, 295, 650–675.

[5] Chen Y., Fan Z.P., Ma J., Zeng S., 2011, A hybrid grouping genetic algorithm for reviewier group construction problem. *Expert Systems with Applications*, 38(3), 2401–2411.

[6] Costa L.R., Aloise D., Mladenović N., 2017, Less is more: basic variable neighborhood search heuristic for balanced minimum sum-of-squares clustering. *Information Sciences*, 415–416, 247–253.

[7] Díaz J.A., Fernández E., Hybrid scatter search and path relinking for the capacitated p-median problem, *European Journal of Operational Research*, 169(2):570–585.

[8] Dhillon I., Guan Y., Kulis B., 2007, Weighted graph cuts without eigenvectors: a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957.

[9] Fan Z.P., Chen Y., Ma J., Zeng S., 2010, A hybrid genetic algorithmic approach to the maximally diverse grouping problem. *Journal of the Operational Research Society*, 62, 92–99.

[10] Feo T.A., Khellaf M., 1990, A class of bounded approximation algorithms for graph partitioning. *Networks*, 20, 181–195.

[11] Gallego M., Laguna M., Martí R., Duarte A., 2013, Tabu search with strategic oscillation for the maximally diverse grouping problem. *Journal of the Operational Research Society*, 64, 724–734.

[12] Glover F., Laguna M., 1997, Tabu search. Springer Science+Business Media New York.

[13] Grötschel M., Wakabayashi Y., 1989, A cutting plane algorithm for a clustering problem. *Mathematical Programming*, 45(1–3), 59–96.

[14] Hao, J.K., 2012, Memetic algorithms in discrete optimization. In: Neri, F., Cotta, C., & Moscato, P. (Eds.), Handbook of Memetic Algorithms, Studies in Computational Intelligence, Vol 379, pp. 73–94, Berlin: Springer.

[15] Hansen P., Mladenović N., 2001, Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3), 449–467.

[16] Johnes J., 2015, Operational Research in education. *European Journal of Operational Research*, 243(3), 683–696.

[17] Krass D., Ovchinnikov A., 2010, Constrained group balancing: Why does it work. *European Journal of Operational Research*, 206(1), 144–154.

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

[19] Lorena N., Antonio L., 2001, Constructive genetic algorithm for clustering problems. *Evolutionary Computation*, 9(3), 309–327.

[20] Lourenco, H.R., Martin, O., & Stützle, T., 2003, Iterated local search. In Glover F., Kochenberger G. (Eds.), Handbook of Metaheuristics, Kluwer.

[21] Palubeckis G., Karčiauskas E., Riškus A., 2011, Comparative performance of three metaheuristic approaches for the maximally diverse grouping problem. *Information Technology and Control*, 40(4), 277–285.

[22] Palubeckis G., Ostreika A., Rubliauskas D., 2015, Maximally diverse grouping: an iterated tabu search approach. *Journal of the Operational Research Society*, 66(4), 579–592.

[23] Rodriguez F.J., Lozano M., García-Martínez C., González-Barrera J.D., 2013, An artificial bee colony algorithm for the maximally diverse grouping problem. *Information Sciences*, 230(1), 183–196.

[24] Singh K., Sundar S., 2019, A new hybrid genetic algorithm for the maximally diverse grouping problem. *International Journal of Machine Learning and Cybernetics*, 10(10), 2921–2940.

[25] Urošević D., 2014, Variable neighborhood search for maximum diverse grouping problem. *Yugoslav Journal of Operations Research*, 24(1), 21–33.

[26] Weitz R., Lakshminarayan S., 1997, An empirical comparison of heuristic and graph theoretic methods for creating maximally diverse groups, VLSI design, and exam scheduling. *Omega*, 25(4), 473–482.

[27] Weitz R., Lakshminarayan S., 1998, An empirical comparison of heuristic methods for creating maximally diverse groups. *Journal of the Operational Research Society*, 49, 635–646.

[28] Yeoh H.K., Nor M.I.M, 2011, An algorithm to form balanced and diverse groups of students. *Computer Applications in Engineering Education*, 19(3), 582–590.

[29] Zhou Y., Hao J.K., Goëffon A., 2016, A three-phased local search approach for the Clique Partitioning problem. *Journal of Combinatorial Optimization*, 32(2), 469–491.

A Appendix

Detailed results of the proposed NDHA algorithm and three main reference algorithms (ITS [22], SGVNS [4], and IMS [18]) on 260 large scale instances with $N = 2000$ or 3000 are summarized in Tables A.1–A.13, where each table corresponds to a subset of benchmarks and the statistical information is the same as in the tables of Section 3. The dominating values are indicated in bold.

One observes from the tables that the NDHA algorithm outperforms significantly the reference algorithms for most instances. Specifically, for the 40 instances with $m = 10$, NDHA outperforms significantly ITS and SGVNS, but performs worse than IMS. For the remaining 220 instances with $m \geq 25$, NDHA outperforms consistently the reference algorithms on all the instances both in terms of f_{best} and f_{avg}.

These outcomes show that the proposed algorithm is particularly suitable to solve large scale instances with a high number of groups, which can be attributed to the neighborhood decomposition strategy.

Table A.1
Comparison of the proposed NDHA algorithm with three best performing algorithms in the literature on the 20 DGS instances with $N = 2000$ and $m = 10$.

Instance				f_{best}				f_{avg}			
Graph	m	L_g	U_g	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
MDG-a_21	10	173	227	1134497	1132526	**1136103**	1135481	1132532.50	1131823.10	**1135473.10**	1134983.70
MDG-a_22	10	173	227	1134114	1132295	**1136012**	1135804	1132404.25	1131546.70	**1135372.65**	1134850.60
MDG-a_23	10	173	227	1133844	1132177	**1135619**	1134911	1131891.95	1131196.85	**1134929.15**	1134280.15
MDG-a_24	10	173	227	1134032	1132845	**1135755**	1135656	1132565.55	1131800.95	**1135180.05**	1134789.45
MDG-a_25	10	173	227	1134484	1133374	**1136116**	1135796	1133109.80	1131986.55	**1135711.15**	1135100.55
MDG-a_26	10	173	227	1134022	1132629	**1135639**	1135322	1132493.10	1131545.60	**1135210.90**	1134695.65
MDG-a_27	10	173	227	1134059	1131812	**1135442**	1134788	1132238.85	1131179.85	**1134887.40**	1134297.25
MDG-a_28	10	173	227	1134124	1132340	**1135850**	1135321	1132573.20	1131694.80	**1135243.55**	1134804.40
MDG-a_29	10	173	227	1134559	1132379	**1136028**	1135795	1132763.35	1131679.00	**1135465.20**	1135145.20
MDG-a_30	10	173	227	1133787	1132089	1135581	**1135698**	1132350.70	1131414.10	**1135191.25**	1134732.95
MDG-a_31	10	173	227	1134796	1133201	**1136262**	1136009	1133220.00	1132288.45	**1135852.45**	1135284.50
MDG-a_32	10	173	227	1134134	1132426	**1136130**	1135294	1132853.80	1131803.05	**1135334.00**	1134854.95
MDG-a_33	10	173	227	1134379	1132133	**1135697**	1135252	1132528.40	1131482.25	**1135195.15**	1134750.05
MDG-a_34	10	173	227	1134612	1132683	**1136216**	1135611	1132670.45	1131854.10	**1135482.25**	1135022.75
MDG-a_35	10	173	227	1134428	1132418	**1135600**	1135199	1132470.80	1131415.65	**1135184.80**	1134624.95
MDG-a_36	10	173	227	1134185	1132439	**1135590**	1135228	1132637.85	1131573.65	**1135172.10**	1134718.30
MDG-a_37	10	173	227	1134409	1132887	**1136081**	1135900	1132852.95	1132014.60	**1135761.55**	1135168.50
MDG-a_38	10	173	227	1134872	1132731	**1135877**	1135483	1132950.45	1131962.60	**1135579.00**	1134810.20
MDG-a_39	10	173	227	1133887	1132362	**1135753**	1135054	1132222.75	1131478.25	**1135090.60**	1134556.85
MDG-a_40	10	173	227	1134405	1133596	**1136407**	1136253	1133177.25	1132386.00	**1135940.20**	1135624.25
Avg.				1134281.45	1132567.1	**1135887.9**	1135492.75	1132625.40	1131706.31	**1135362.83**	1134854.76
#Best				0	0	**19**	1	0	0	**20**	0
p-value				8.86E-5	8.86E-5	1.20E-4		8.86E-5	8.86E-5	8.86E-5	

Table A.2
Comparison of the proposed NDHA algorithm with three best performing algorithms in the literature on the 20 DGS instances with $N = 2000$ and $m = 25$.

Instance				f_{best}				f_{avg}			
Graph	m	L_g	U_g	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
MDG-a_21	25	51	109	539434	539367	541212	**541440**	537963.35	539000.30	540796.65	**540969.30**
MDG-a_22	25	51	109	539663	539468	541228	**541528**	538041.20	538928.40	540743.35	**540934.40**
MDG-a_23	25	51	109	539029	539587	540912	**540926**	538021.45	539144.85	540582.10	**540679.00**
MDG-a_24	25	51	109	539255	539469	541109	**541327**	538229.35	539177.80	540727.90	**540901.10**
MDG-a_25	25	51	109	539655	539599	541393	**541558**	538479.70	539176.60	540816.70	**540997.05**
MDG-a_26	25	51	109	539802	539669	541037	**541330**	538169.10	539097.30	540676.75	**540870.75**
MDG-a_27	25	51	109	539369	539853	541003	**541159**	538022.30	539179.35	540571.55	**540593.35**
MDG-a_28	25	51	109	539281	539352	541245	**541319**	538174.10	539064.00	540832.05	**540953.25**
MDG-a_29	25	51	109	539280	539772	541121	**541937**	538038.70	539187.10	540849.65	**541031.65**
MDG-a_30	25	51	109	539446	539709	540945	**541355**	538276.55	539238.40	540708.60	**540900.20**
MDG-a_31	25	51	109	539592	539383	541500	**541590**	538547.15	538950.80	541052.65	**541200.00**
MDG-a_32	25	51	109	539245	539575	541319	**541403**	538148.05	538950.40	540848.20	**540911.00**
MDG-a_33	25	51	109	539300	540021	541281	**541390**	538163.65	539453.30	540785.25	**540838.80**
MDG-a_34	25	51	109	539614	539633	541267	**541445**	538279.75	539105.90	540910.80	**540971.20**
MDG-a_35	25	51	109	539028	539533	**541365**	541113	538195.45	539071.55	540651.85	**540766.10**
MDG-a_36	25	51	109	539404	539829	541259	**541448**	538129.85	539196.80	540816.40	**540917.90**
MDG-a_37	25	51	109	539705	539402	541203	**541477**	538489.40	538913.80	540942.55	**541078.80**
MDG-a_38	25	51	109	539646	539454	541210	**541390**	538295.95	539002.80	540872.65	**541109.40**
MDG-a_39	25	51	109	539376	539794	541237	**541397**	538245.60	539169.05	540769.70	**540857.30**
MDG-a_40	25	51	109	539917	539268	541498	**541605**	538651.95	538960.80	541110.25	**541247.60**
Avg.				539452.05	539586.85	541217.2	**541406.85**	538228.13	539098.47	540803.28	**540936.41**
#Best				0	0	1	19	0	0	0	**20**
p-value				8.86E-5	8.86E-5	7.79E-4		8.86E-5	8.86E-5	8.86E-5	

Table A.3
Comparison of the proposed NDHA algorithm with three best performing algorithms in the literature on the 20 DGS instances with $N = 2000$, $m = 50$, $L_g = 26$, and $U_g = 54$.

Instance				f_{best}				f_{avg}			
Graph	m	L_g	U_g	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
MDG-a_21	50	26	54	291135	291797	292570	**293232**	289892.35	291521.00	292266.60	**292879.60**
MDG-a_22	50	26	54	290930	292077	292639	**293145**	289950.60	291360.55	292342.05	**292839.15**
MDG-a_23	50	26	54	290880	291761	292656	**293163**	289878.55	291388.90	292249.15	**292825.15**
MDG-a_24	50	26	54	290833	291772	292610	**293228**	290023.55	291465.85	292348.40	**292790.30**
MDG-a_25	50	26	54	290863	291799	292669	**293106**	290009.20	291488.55	292360.55	**292822.90**
MDG-a_26	50	26	54	290941	291830	292582	**293199**	289984.45	291420.80	292383.55	**292883.90**
MDG-a_27	50	26	54	290800	291728	292500	**293047**	289963.85	291248.55	292135.70	**292710.00**
MDG-a_28	50	26	54	291185	291779	292608	**293065**	289985.20	291464.50	292334.35	**292855.60**
MDG-a_29	50	26	54	290879	291684	292625	**293149**	289818.45	291468.05	292336.65	**292841.30**
MDG-a_30	50	26	54	291069	291825	292572	**293212**	289877.05	291488.90	292299.15	**292838.55**
MDG-a_31	50	26	54	290912	291874	292658	**293294**	289873.00	291527.85	292380.95	**292900.30**
MDG-a_32	50	26	54	291030	291873	292621	**293192**	289939.35	291433.70	292276.30	**292887.95**
MDG-a_33	50	26	54	291093	291873	292575	**293251**	289962.40	291474.50	292274.70	**292868.15**
MDG-a_34	50	26	54	290916	291792	292588	**293237**	290048.25	291472.70	292306.25	**292892.85**
MDG-a_35	50	26	54	290975	291777	292549	**293042**	289846.70	291360.15	292278.35	**292745.15**
MDG-a_36	50	26	54	290934	291824	292786	**293069**	289829.40	291445.05	292321.40	**292765.30**
MDG-a_37	50	26	54	290865	291923	292682	**293266**	289720.10	291445.45	292416.60	**292881.35**
MDG-a_38	50	26	54	291224	291793	292558	**293279**	290074.35	291448.25	292260.35	**292889.90**
MDG-a_39	50	26	54	291199	291675	292445	**293149**	289973.25	291371.40	292198.10	**292824.15**
MDG-a_40	50	26	54	291445	291978	292690	**293448**	290465.35	291543.10	292398.70	**293002.90**
Avg.				291005.4	291821.7	292609.15	**293188.65**	289955.77	291441.89	292308.39	**292847.22**
#Best				0	0	0	**20**	0	0	0	**20**
p-value				8.86E-5	8.86E-5	8.86E-5		8.86E-5	8.86E-5	8.86E-5	

Table A.4
Comparison of the proposed NDHA algorithm with three best performing algorithms in the literature on the 20 DGS instances with $N = 2000$, $m = 50$, $L_g = 32$, and $U_g = 48$.

Instance				f_{best}				f_{avg}			
Graph	m	L_g	U_g	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
MDG-a_21	50	32	48	273075	273549	274369	**275062**	271766.95	273231.80	274077.45	**274521.55**
MDG-a_22	50	32	48	272771	273301	274424	**274726**	271802.65	273077.60	274016.55	**274455.60**
MDG-a_23	50	32	48	272642	273542	274178	**274692**	271655.35	273060.55	273960.50	**274382.85**
MDG-a_24	50	32	48	272816	273494	274356	**275046**	271739.15	273151.55	274053.35	**274524.35**
MDG-a_25	50	32	48	272648	273636	274308	**274867**	271714.55	273242.20	274057.75	**274515.20**
MDG-a_26	50	32	48	272616	273493	274359	**274757**	271688.85	273215.15	274086.05	**274449.50**
MDG-a_27	50	32	48	272859	273470	274379	**274932**	271686.80	273232.30	273851.35	**274387.60**
MDG-a_28	50	32	48	272958	273608	274326	**274850**	271763.90	273262.75	274040.20	**274469.70**
MDG-a_29	50	32	48	272811	273648	274548	**274865**	271917.35	273270.60	273995.85	**274482.25**
MDG-a_30	50	32	48	272880	273458	274394	**274886**	271720.85	273218.20	274011.90	**274440.35**
MDG-a_31	50	32	48	272900	273650	274381	**274955**	271954.25	273267.30	274154.75	**274599.30**
MDG-a_32	50	32	48	273085	273447	274307	**275001**	271987.25	273252.00	274044.60	**274533.15**
MDG-a_33	50	32	48	272988	273427	274337	**275028**	271703.55	273172.95	274049.40	**274434.35**
MDG-a_34	50	32	48	273152	273512	274263	**274819**	272066.65	273240.30	274012.60	**274450.85**
MDG-a_35	50	32	48	273030	273358	274257	**274751**	271968.15	273156.15	273959.90	**274371.15**
MDG-a_36	50	32	48	272844	273606	274605	**274797**	271678.40	273250.20	274024.95	**274493.65**
MDG-a_37	50	32	48	272854	273465	274684	**274983**	271607.20	273265.70	274123.35	**274596.25**
MDG-a_38	50	32	48	272654	273629	274476	**274902**	271875.50	273344.05	274038.35	**274569.85**
MDG-a_39	50	32	48	272582	273439	274320	**274909**	271555.15	273135.55	274024.25	**274543.10**
MDG-a_40	50	32	48	272868	273722	274675	**275142**	271900.55	273385.55	274199.20	**274905.75**
Avg.				272851.65	273522.70	274397.30	**274898.50**	271787.65	273221.62	274039.12	**274506.32**
#Best				0	0	0	**20**	0	0	0	**20**
p-value				8.86E-5	8.86E-5	8.86E-5		8.86E-5	8.86E-5	8.86E-5	

Table A.5
Comparison of the proposed NDHA algorithm with three best performing algorithms in the literature on the 20 DGS instances with $N = 2000$ and $m = 100$.

Instance				f_{best}				f_{avg}			
Graph	m	L_g	U_g	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
MDG-a_21	100	13	27	159026	159253	159769	**160823**	158096.95	158992.15	159519.45	**160567.00**
MDG-a_22	100	13	27	158616	159218	159690	**160869**	157851.50	158985.20	159526.55	**160535.65**
MDG-a_23	100	13	27	158736	159265	159777	**160676**	157860.40	159022.55	159550.35	**160484.10**
MDG-a_24	100	13	27	158592	159362	159791	**160812**	157715.20	159043.40	159605.40	**160522.40**
MDG-a_25	100	13	27	158750	159293	159799	**160673**	157913.45	159090.75	159519.00	**160525.65**
MDG-a_26	100	13	27	158653	159326	159933	**160887**	157806.25	159075.35	159669.05	**160575.35**
MDG-a_27	100	13	27	158697	159191	159717	**160726**	157841.85	158967.00	159494.30	**160522.10**
MDG-a_28	100	13	27	158674	159356	159756	**160737**	157837.75	159058.85	159552.60	**160544.20**
MDG-a_29	100	13	27	158809	159324	159823	**160792**	157808.85	159002.85	159498.25	**160574.85**
MDG-a_30	100	13	27	158650	159468	159883	**160801**	157912.95	159073.45	159705.40	**160596.75**
MDG-a_31	100	13	27	158778	159289	159709	**160861**	157998.75	159037.70	159569.15	**160587.30**
MDG-a_32	100	13	27	158697	159309	159751	**160810**	157853.35	158977.75	159574.90	**160519.70**
MDG-a_33	100	13	27	158529	159225	159822	**160835**	157793.35	158969.25	159531.75	**160578.40**
MDG-a_34	100	13	27	158800	159300	159747	**160830**	158047.25	159024.25	159547.50	**160565.20**
MDG-a_35	100	13	27	158822	159400	160003	**160715**	157707.45	158999.35	159537.25	**160443.85**
MDG-a_36	100	13	27	158604	159266	159833	**160902**	158026.75	159059.45	159511.70	**160511.85**
MDG-a_37	100	13	27	158731	159257	159889	**160883**	157920.80	159047.10	159605.55	**160609.60**
MDG-a_38	100	13	27	158670	159282	159804	**160899**	157882.70	158985.55	159456.50	**160563.75**
MDG-a_39	100	13	27	158821	159275	159821	**160909**	157915.40	158905.95	159547.45	**160534.95**
MDG-a_40	100	13	27	159247	159283	159915	**160837**	158160.50	158982.10	159636.60	**160628.80**
Avg.				158745.1	159297.1	159811.6	**160813.85**	157897.57	159015.00	159557.94	**160549.57**
#Best				0	0	0	**20**	0	0	0	**20**
p-value				8.86E-5	8.86E-5	8.86E-5		8.86E-5	8.86E-5	8.86E-5	

Table A.6
Comparison of the proposed NDHA algorithm with three best performing algorithms in the literature on the 20 DGS instances with $N = 2000$ and $m = 200$.

Instance				f_{best}				f_{avg}			
Graph	m	L_g	U_g	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
MDG-a_21	200	6	14	88227	88435	89009	**89673**	87785.55	88295.80	88771.15	**89453.65**
MDG-a_22	200	6	14	88161	88479	89217	**89632**	87609.05	88276.80	88873.15	**89391.70**
MDG-a_23	200	6	14	88228	88400	88656	**89582**	87766.40	88288.55	88485.60	**89406.30**
MDG-a_24	200	6	14	88282	88388	88739	**89575**	87707.25	88276.00	88487.15	**89380.30**
MDG-a_25	200	6	14	88351	88627	88717	**89561**	87906.80	88392.70	88526.30	**89416.20**
MDG-a_26	200	6	14	88472	88686	88851	**89630**	87722.55	88465.95	88548.65	**89438.55**
MDG-a_27	200	6	14	88282	88515	88735	**89482**	87776.85	88392.50	88570.65	**89380.60**
MDG-a_28	200	6	14	88301	88495	88703	**89609**	87655.00	88324.35	88383.40	**89388.75**
MDG-a_29	200	6	14	88242	88623	88716	**89552**	87726.45	88347.70	88494.75	**89416.05**
MDG-a_30	200	6	14	88477	88414	88741	**89644**	87824.95	88298.25	88470.25	**89447.45**
MDG-a_31	200	6	14	88243	88532	88771	**89580**	87673.80	88337.70	88571.65	**89456.40**
MDG-a_32	200	6	14	88243	88458	88692	**89520**	87674.85	88284.35	88514.60	**89410.45**
MDG-a_33	200	6	14	88319	88544	88620	**89539**	87705.60	88329.45	88462.60	**89424.20**
MDG-a_34	200	6	14	88272	88543	88897	**89704**	87658.15	88317.65	88706.85	**89435.05**
MDG-a_35	200	6	14	88333	88514	88816	**89589**	87742.30	88271.35	88591.20	**89419.70**
MDG-a_36	200	6	14	88329	88531	88898	**89588**	87743.50	88337.80	88690.15	**89425.75**
MDG-a_37	200	6	14	88282	88453	88917	**89611**	87843.50	88292.25	88708.75	**89388.40**
MDG-a_38	200	6	14	88325	88520	88830	**89676**	87678.55	88313.90	88717.60	**89465.00**
MDG-a_39	200	6	14	88344	88530	88887	**89578**	87742.95	88342.25	88642.95	**89401.25**
MDG-a_40	200	6	14	88364	88557	89180	**89644**	87733.00	88339.10	88781.05	**89445.40**
Avg.				88303.85	88512.20	88829.60	**89598.45**	87733.85	88326.22	88599.92	**89419.56**
#Best				0	0	0	**20**	0	0	0	**20**
p-value				8.86E-5	8.86E-5	8.86E-5		8.86E-5	8.86E-5	8.86E-5	

Table A.7
Comparison of the proposed NDHA algorithm with three best performing algorithms in the literature on the 20 EGS instances with $N = 2000$ and $m = 10$.

Instance				f_{best}				f_{avg}			
Graph	m	L_g	U_g	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
MDG-a_21	10	200	200	1115670	1114042	**1117329**	1116899	1113959.45	1113305.90	**1116979.60**	1116262.35
MDG-a_22	10	200	200	1115321	1113721	1117023	**1117135**	1113665.70	1113035.30	**1116590.15**	1116258.00
MDG-a_23	10	200	200	1114892	1113478	1116532	**1116772**	1113272.70	1112722.85	**1116179.45**	1115769.35
MDG-a_24	10	200	200	1115371	1113662	**1117108**	1116750	1114181.95	1113007.90	**1116631.90**	1116139.75
MDG-a_25	10	200	200	1115958	1114335	**1117273**	1117265	1114653.20	1113459.75	**1116971.70**	1116676.25
MDG-a_26	10	200	200	1115590	1113747	**1116968**	1116706	1114141.50	1112874.85	**1116482.35**	1116062.35
MDG-a_27	10	200	200	1114931	1113363	**1116630**	1116189	1113642.65	1112788.20	**1116158.30**	1115640.35
MDG-a_28	10	200	200	1115613	1113878	**1117185**	1116643	1113922.30	1112992.90	**1116613.20**	1116161.85
MDG-a_29	10	200	200	1116073	1113935	**1117251**	1117083	1114258.00	1113218.95	**1116856.55**	1116448.40
MDG-a_30	10	200	200	1115358	1113918	**1117019**	1116680	1113618.85	1112977.00	**1116586.80**	1116015.40
MDG-a_31	10	200	200	1115889	1114290	**1117450**	1117354	1114371.55	1113477.65	**1117122.05**	1116780.00
MDG-a_32	10	200	200	1115098	1113951	**1117229**	1117154	1113961.90	1113096.55	**1116816.80**	1116506.95
MDG-a_33	10	200	200	1115074	1113867	**1117055**	1116663	1113714.90	1112992.65	**1116508.75**	1116155.00
MDG-a_34	10	200	200	1115691	1114120	**1117447**	1116856	1114448.65	1113533.95	**1116961.50**	1116390.50
MDG-a_35	10	200	200	1115086	1114196	**1116854**	1116674	1113983.25	1112705.70	**1116383.20**	1116028.95
MDG-a_36	10	200	200	1115427	1113669	**1117048**	1117042	1113749.60	1113041.15	**1116601.60**	1116181.85
MDG-a_37	10	200	200	1115800	1114510	**1117581**	1117109	1114503.25	1113459.35	**1117121.45**	1116566.40
MDG-a_38	10	200	200	1115525	1114085	**1117678**	1116862	1114260.05	1113419.40	**1116953.85**	1116478.20
MDG-a_39	10	200	200	1114913	1113545	**1116910**	1116677	1113737.60	1112885.10	**1116443.75**	1116072.50
MDG-a_40	10	200	200	1116449	1114319	**1117864**	1117388	1115060.00	1113845.05	**1117383.45**	1116793.60
Avg.				1115486.45	1113931.55	**1117171.70**	1116895.05	1114055.35	1113142.01	**1116717.32**	1116269.40
#Best				0	0	**18**	2	0	0	**20**	0
p-value				8.86E-5	8.86E-5	6.81E-4		8.86E-5	8.86E-5	8.86E-5	

Table A.8
Comparison of the proposed NDHA algorithm with three best performing algorithms in the literature on the 20 EGS instances with $N = 2000$ and $m = 25$.

Instance				f_{best}				f_{avg}			
Graph	m	L_g	U_g	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
MDG-a_21	25	80	80	486111	486385	487689	**488046**	484925.90	485743.55	487282.20	**487697.75**
MDG-a_22	25	80	80	485737	486122	487629	**487882**	484514.45	485580.50	487195.05	**487503.65**
MDG-a_23	25	80	80	485651	485893	487264	**487752**	484595.35	485455.50	487102.30	**487335.90**
MDG-a_24	25	80	80	485919	486155	487396	**488040**	484593.65	485735.30	487171.20	**487511.40**
MDG-a_25	25	80	80	485830	486233	487615	**487828**	484923.30	485735.05	487331.60	**487540.75**
MDG-a_26	25	80	80	485830	486091	487544	**488012**	484753.95	485611.05	487196.35	**487477.65**
MDG-a_27	25	80	80	485604	486080	487376	**487599**	484393.25	485506.55	487055.50	**487307.15**
MDG-a_28	25	80	80	486129	486185	487553	**487899**	484725.15	485574.80	487207.60	**487492.10**
MDG-a_29	25	80	80	486329	486033	487548	**487971**	484994.10	485656.95	487291.10	**487595.40**
MDG-a_30	25	80	80	486017	485896	487361	**487785**	484673.45	485599.15	487183.90	**487491.15**
MDG-a_31	25	80	80	486137	486545	487885	**488002**	484986.05	485894.60	487513.05	**487662.40**
MDG-a_32	25	80	80	486174	486130	487588	**487911**	484626.85	485589.55	487264.25	**487581.95**
MDG-a_33	25	80	80	485821	485975	487582	**488113**	484673.75	485611.85	487231.65	**487547.90**
MDG-a_34	25	80	80	486631	486286	487781	**488012**	484974.30	485640.30	487284.50	**487624.45**
MDG-a_35	25	80	80	485749	485912	487442	**487954**	484481.75	485520.30	487183.75	**487484.70**
MDG-a_36	25	80	80	486013	486102	487510	**487950**	484752.70	485601.05	487105.30	**487555.20**
MDG-a_37	25	80	80	485925	486265	487705	**487925**	484918.05	485906.80	487370.60	**487684.10**
MDG-a_38	25	80	80	485897	486129	487657	**487887**	484894.45	485762.45	487276.05	**487529.15**
MDG-a_39	25	80	80	485947	486033	487465	**487848**	484797.10	485607.90	487146.40	**487438.85**
MDG-a_40	25	80	80	486551	486745	487918	**488055**	485135.80	485934.65	487463.60	**487702.90**
Avg.				486000.1	486159.75	487575.4	**487923.55**	484766.67	485663.39	487242.80	**487538.23**
#Best				0	0	0	**20**	0	0	0	**20**
p-value				8.86E-5	8.86E-5	8.86E-5		8.86E-5	8.86E-5	8.86E-5	

Table A.9
Comparison of the proposed NDHA algorithm with three best performing algorithms in the literature on the 20 EGS instances with $N = 2000$ and $m = 50$.

Instance				f_{best}				f_{avg}			
Graph	m	L_g	U_g	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
MDG-a_21	50	40	40	263512	264642	265571	**265951**	262625.40	264329.35	265192.95	**265634.50**
MDG-a_22	50	40	40	263876	264580	265437	**265832**	262750.50	264281.15	265046.65	**265438.90**
MDG-a_23	50	40	40	263731	264493	265316	**265884**	262651.15	264156.50	265065.25	**265504.10**
MDG-a_24	50	40	40	263736	264575	265422	**265956**	262500.25	264290.55	265093.25	**265573.55**
MDG-a_25	50	40	40	263688	264556	265354	**265866**	262863.60	264287.50	265134.10	**265519.90**
MDG-a_26	50	40	40	263837	264462	265482	**265933**	262679.60	264225.70	265124.95	**265609.55**
MDG-a_27	50	40	40	263603	264517	265474	**265928**	262602.80	264116.65	265034.55	**265415.35**
MDG-a_28	50	40	40	263628	264548	265316	**265995**	262730.25	264199.10	265081.20	**265573.35**
MDG-a_29	50	40	40	264015	264709	265554	**265858**	262833.90	264272.75	265169.55	**265589.55**
MDG-a_30	50	40	40	263732	264509	265467	**266144**	262672.25	264240.95	265173.15	**265558.55**
MDG-a_31	50	40	40	263744	264532	265542	**266022**	262728.60	264349.25	265219.85	**265616.70**
MDG-a_32	50	40	40	263822	264579	265557	**265866**	262668.55	264277.00	265132.95	**265571.40**
MDG-a_33	50	40	40	263688	264426	265462	**265948**	262664.90	264224.65	265100.90	**265601.60**
MDG-a_34	50	40	40	263561	264832	265387	**265927**	262645.30	264365.55	265108.75	**265592.85**
MDG-a_35	50	40	40	263555	264518	265334	**266087**	262578.05	264219.60	265069.20	**265465.35**
MDG-a_36	50	40	40	263895	264451	265316	**265782**	262646.85	264228.90	265074.20	**265514.10**
MDG-a_37	50	40	40	263714	264499	265308	**265911**	262717.20	264251.10	265079.35	**265642.75**
MDG-a_38	50	40	40	263554	264623	265600	**265839**	262592.60	264342.35	265209.95	**265506.30**
MDG-a_39	50	40	40	263657	264603	265314	**265810**	262655.90	264236.85	265055.75	**265503.35**
MDG-a_40	50	40	40	263841	264576	265546	**266324**	262776.30	264363.00	265228.50	**265769.55**
Avg.				263719.45	264561.5	265437.95	**265943.15**	262679.20	264262.92	265119.75	**265560.06**
#Best				0	0	0	**20**	0	0	0	**20**
p-value				8.86E-5	8.86E-5	8.86E-5		8.86E-5	8.86E-5	8.86E-5	

Table A.10
Comparison of the proposed NDHA algorithm with three best performing algorithms in the literature on the 20 EGS instances with $N = 2000$ and $m = 100$.

Instance				f_{best}				f_{avg}			
Graph	m	L_g	U_g	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
MDG-a_21	100	20	20	143675	144813	144748	**145401**	143083.65	144359.10	144563.20	**145170.35**
MDG-a_22	100	20	20	143933	144525	144847	**145432**	143077.45	144316.00	144613.45	**145177.90**
MDG-a_23	100	20	20	143738	144554	144753	**145261**	142909.50	144314.15	144531.45	**145057.25**
MDG-a_24	100	20	20	144003	144536	144740	**145593**	143076.35	144293.60	144526.75	**145155.65**
MDG-a_25	100	20	20	144161	144473	144965	**145460**	143044.80	144324.10	144599.40	**145155.40**
MDG-a_26	100	20	20	143913	144555	144854	**145441**	143093.70	144373.30	144609.65	**145211.55**
MDG-a_27	100	20	20	143910	144558	144762	**145435**	143150.65	144286.60	144535.40	**145076.95**
MDG-a_28	100	20	20	144043	144580	144882	**145444**	143087.50	144338.95	144591.10	**145154.00**
MDG-a_29	100	20	20	143836	144699	144795	**145468**	143146.15	144364.65	144606.20	**145164.40**
MDG-a_30	100	20	20	143978	144613	144830	**145873**	143048.50	144364.55	144580.50	**145171.35**
MDG-a_31	100	20	20	143876	144684	144930	**145548**	143088.35	144369.70	144637.50	**145186.65**
MDG-a_32	100	20	20	143878	144576	144810	**145477**	143030.35	144357.60	144562.30	**145137.75**
MDG-a_33	100	20	20	143957	144873	144886	**145319**	143070.75	144414.75	144628.60	**145123.95**
MDG-a_34	100	20	20	143917	144570	144905	**145476**	143145.65	144310.85	144647.30	**145191.00**
MDG-a_35	100	20	20	143901	144501	144768	**145314**	143147.05	144276.10	144512.55	**145134.10**
MDG-a_36	100	20	20	143853	144560	144898	**145458**	143097.65	144290.55	144603.10	**145150.85**
MDG-a_37	100	20	20	144108	144534	144840	**145394**	143171.00	144311.95	144654.55	**145166.30**
MDG-a_38	100	20	20	143725	144706	144867	**145487**	143170.05	144334.90	144567.00	**145178.35**
MDG-a_39	100	20	20	143803	144534	144760	**145609**	143081.45	144244.95	144553.90	**145210.40**
MDG-a_40	100	20	20	144138	144605	144850	**145648**	143057.20	144380.00	144602.95	**145252.25**
Avg.				143917.3	144602.45	144834.5	**145476.9**	143088.89	144331.32	144586.34	**145161.32**
#Best				0	0	0	**20**	0	0	0	**20**
p-value				8.86E-5	8.86E-5	8.86E-5		8.86E-5	8.86E-5	8.86E-5	

Table A.11
Comparison of the proposed NDHA algorithm with three best performing algorithms in the literature on the 20 EGS instances with $N = 2000$ and $m = 200$.

Instance				f_{best}				f_{avg}			
Graph	m	L_g	U_g	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
MDG-a_21	200	10	10	77065	76803	77127	**78193**	76530.95	76652.45	77030.05	**78101.00**
MDG-a_22	200	10	10	76971	76850	77176	**78423**	76469.25	76690.10	77039.00	**78098.35**
MDG-a_23	200	10	10	77002	76847	77231	**78253**	76372.40	76684.90	77071.70	**78111.00**
MDG-a_24	200	10	10	76933	76833	77211	**78300**	76515.95	76686.15	77055.40	**78075.35**
MDG-a_25	200	10	10	77198	76835	77245	**78266**	76561.05	76681.10	77039.70	**78143.55**
MDG-a_26	200	10	10	76859	76804	77313	**78324**	76497.85	76706.05	77078.90	**78107.90**
MDG-a_27	200	10	10	76875	76841	77269	**78220**	76412.85	76642.40	77024.35	**78085.00**
MDG-a_28	200	10	10	76853	76809	77353	**78208**	76438.00	76693.25	77064.80	**78107.75**
MDG-a_29	200	10	10	76942	76879	77212	**78271**	76489.25	76706.70	77058.25	**78104.90**
MDG-a_30	200	10	10	77011	76890	77165	**78187**	76513.90	76720.40	77068.60	**78092.05**
MDG-a_31	200	10	10	76967	76839	77222	**78380**	76480.75	76687.70	77052.85	**78255.45**
MDG-a_32	200	10	10	77005	76796	77225	**78252**	76399.50	76669.85	77020.20	**78117.95**
MDG-a_33	200	10	10	76981	76862	77182	**78234**	76474.45	76722.15	77072.60	**78085.05**
MDG-a_34	200	10	10	77116	76927	77166	**78193**	76521.45	76710.90	77041.80	**78082.05**
MDG-a_35	200	10	10	77018	76853	77195	**78332**	76454.05	76681.45	77040.15	**78094.25**
MDG-a_36	200	10	10	76880	76801	77240	**78348**	76404.25	76702.35	77052.80	**78158.55**
MDG-a_37	200	10	10	76913	76840	77211	**78335**	76484.70	76693.20	77051.05	**78126.90**
MDG-a_38	200	10	10	76995	76840	77234	**78189**	76413.50	76681.05	77038.40	**78100.00**
MDG-a_39	200	10	10	77009	76803	77232	**78290**	76437.25	76667.80	77025.15	**78122.50**
MDG-a_40	200	10	10	76865	76882	77230	**78340**	76488.25	76713.95	77086.05	**78155.55**
Avg.				76972.90	76841.70	77221.95	**78276.90**	76467.98	76689.70	77050.59	**78116.26**
#Best				0	0	0	**20**	0	0	0	**20**
p-value				8.86E-5	8.86E-5	8.86E-5		8.86E-5	8.86E-5	8.86E-5	

Table A.12
Comparison of the proposed NDHA algorithm with three best performing algorithms in the literature on the 20 DGS instances with $N = 3000$ and $m = 50$.

Instance				f_{best}				f_{avg}			
Graph	m	L_g	U_g	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
MDG-c_1	50	48	72	57921457	58093235	58265192	**58340690**	57737678.30	58035513.50	58209687.40	**58297673.25**
MDG-c_2	50	48	72	57902133	58099262	58257908	**58337196**	57731383.75	58026098.05	58209556.50	**58299371.40**
MDG-c_3	50	48	72	57934848	58055836	58241554	**58311192**	57762546.30	58014337.00	58191269.80	**58270396.10**
MDG-c_4	50	48	72	57927582	58037524	58233772	**58329732**	57764219.85	57998056.85	58194485.50	**58276632.95**
MDG-c_5	50	48	72	57923652	58055692	58214064	**58328164**	57759631.00	57994387.20	58170320.15	**58274835.70**
MDG-c_6	50	48	72	57864117	58016049	58224119	**58309892**	57721643.20	57986567.55	58168981.80	**58256842.00**
MDG-c_7	50	48	72	57863533	58034097	58234312	**58313700**	57698298.50	57980907.15	58186693.90	**58280775.25**
MDG-c_8	50	48	72	57934738	58045305	58209856	**58288736**	57730173.20	58005747.75	58174878.90	**58257575.90**
MDG-c_9	50	48	72	57910127	58017357	58218441	**58319076**	57735321.20	57983943.75	58163082.30	**58264473.95**
MDG-c_10	50	48	72	57930587	57997933	58201426	**58294490**	57736078.60	57965143.80	58174383.15	**58264257.90**
MDG-c_11	50	48	72	57903371	58057105	58269895	**58336603**	57719528.20	58007027.90	58227824.85	**58286386.60**
MDG-c_12	50	48	72	57900418	58081540	58230643	**58330830**	57707521.30	58025359.15	58184171.95	**58281631.20**
MDG-c_13	50	48	72	57858714	58083409	58269976	**58348524**	57716409.80	58031488.85	58202546.90	**58279226.65**
MDG-c_14	50	48	72	57882617	58038649	58289363	**58300407**	57721202.00	57984240.60	58202482.10	**58255428.60**
MDG-c_15	50	48	72	57931495	58048003	58243022	**58329292**	57746547.60	57985029.30	58197244.85	**58265913.85**
MDG-c_16	50	48	72	57960453	58059638	58223406	**58336695**	57760872.90	58010567.10	58187356.65	**58294369.60**
MDG-c_17	50	48	72	57918754	58029359	58213613	**58320502**	57741845.75	57983441.85	58163916.65	**58269475.80**
MDG-c_18	50	48	72	57884836	58003403	58190902	**58311686**	57700437.65	57963090.00	58153413.50	**58253890.75**
MDG-c_19	50	48	72	57886748	58031546	58224417	**58371723**	57709285.60	57998606.85	58189788.00	**58281675.95**
MDG-c_20	50	48	72	57903276	58071049	58212202	**58334988**	57748397.25	58002919.25	58171944.40	**58272124.90**
Avg.				57907172.80	58047799.55	58233404.15	**58324705.90**	57732451.10	57999123.67	58186201.46	**58274147.92**
#Best				0	0	0	**20**	0	0	0	**20**
p-value				8.86E-5	8.86E-5	8.86E-5		8.86E-5	8.86E-5	8.86E-5	

Table A.13
Comparison of the proposed NDHA algorithm with three best performing algorithms in the literature on the 20 EGS instances with $N = 3000$ and $m = 50$.

Instance				f_{best}				f_{avg}			
Graph	m	L_g	U_g	**ITS**	**SGVNS**	**IMS**	**NDHA**	**ITS**	**SGVNS**	**IMS**	**NDHA**
MDG-c_1	50	60	60	55935354	56066732	56300952	**56367170**	55773172.85	56026456.90	56261757.35	**56333433.90**
MDG-c_2	50	60	60	55935048	56052085	56304407	**56391411**	55751718.00	56012225.35	56264734.25	**56345921.00**
MDG-c_3	50	60	60	55876810	55928266	56271740	**56368135**	55757466.30	55886719.80	56233584.00	**56330534.85**
MDG-c_4	50	60	60	55892522	56045917	56274086	**56355625**	55753144.20	56001442.40	56233863.20	**56321400.10**
MDG-c_5	50	60	60	55966544	56025930	56291632	**56346700**	55764810.85	55983690.45	56228552.10	**56313011.30**
MDG-c_6	50	60	60	55872193	56050890	56240478	**56346004**	55757932.70	56008253.65	56210354.00	**56302313.75**
MDG-c_7	50	60	60	55898725	56004130	56284630	**56372432**	55741551.65	55960750.75	56224220.50	**56319651.95**
MDG-c_8	50	60	60	55911548	56031405	56249087	**56362161**	55745946.30	55988150.90	56215130.00	**56320663.10**
MDG-c_9	50	60	60	55871860	55916383	56265244	**56345352**	55762150.35	55870335.40	56211872.45	**56292812.70**
MDG-c_10	50	60	60	55920804	55975884	56271841	**56341222**	55889867.25	55920627.00	56220974.15	**56292495.60**
MDG-c_11	50	60	60	55916984	56079876	56245962	**56382456**	55739899.00	56008914.10	56214501.30	**56326674.85**
MDG-c_12	50	60	60	55895697	56016471	56246744	**56330408**	55752006.35	55942412.60	56215907.70	**56293618.05**
MDG-c_13	50	60	60	55954309	56031145	56275819	**56351898**	55769196.40	55985360.10	56241430.35	**56301577.00**
MDG-c_14	50	60	60	55942413	55986823	56279869	**56355641**	55765380.20	55948686.80	56233195.60	**56299851.20**
MDG-c_15	50	60	60	55917971	55993491	56277686	**56393376**	55753017.35	55945782.20	56229065.85	**56349326.05**
MDG-c_16	50	60	60	55930835	56032237	56281453	**56403871**	55761254.95	55987905.60	56242444.65	**56360680.45**
MDG-c_17	50	60	60	55914349	56015346	56273992	**56393589**	55755871.50	55973726.35	56227222.95	**56337958.45**
MDG-c_18	50	60	60	55882209	56002250	56223804	**56369757**	55730098.30	55970535.10	56193093.10	**56315879.40**
MDG-c_19	50	60	60	55917957	56040091	56269179	**56392641**	55774459.50	56000374.75	56229431.65	**56347320.95**
MDG-c_20	50	60	60	55931252	56030384	56261671	**56369775**	55745101.20	55988408.65	56229092.20	**56314192.25**
Avg.				55914269.20	56016286.80	56269513.80	**56366981.20**	55762202.26	55970537.94	56228021.37	**56320965.85**
#Best				0	0	0	**20**	0	0	0	**20**
p-value				8.86E-5	8.86E-5	8.86E-5		8.86E-5	8.86E-5	8.86E-5	