

# Iterated variable neighborhood search for the capacitated clustering problem

Xiangjing Lai<sup>a</sup> and Jin-Kao Hao<sup>a,b,\*</sup>,

<sup>a</sup>*LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France*

<sup>b</sup>*Institut Universitaire de France, Paris, France*  
*Engineering Applications of Artificial Intelligence*  
<http://dx.doi.org/10.1016/j.engappai.2016.08.004>

---

## Abstract

The NP-hard capacitated clustering problem (CCP) is a general model with a number of relevant applications. This paper proposes a highly effective iterated variable neighborhood search (IVNS) algorithm for solving the problem. IVNS combines an extended variable neighborhood descent method and a randomized shake procedure to explore effectively the search space. The computational results obtained on three sets of 133 benchmarks reveal that the proposed algorithm competes favorably with the state-of-the-art algorithms in the literature both in terms of solution quality and computational efficiency. In particular, IVNS discovers an improved best known result (new lower bounds) for 28 out of 83 most popular instances, while matching the current best known results for the remaining 55 instances. Several essential components of the proposed algorithm are investigated to understand their impacts on the performance of algorithm.

**Keywords:** Capacitated clustering; grouping problem; variable neighborhood search; heuristics.

---

## 1 Introduction

2 Given a weighted undirect graph  $G = (V, E, C, w)$ , where  $V = \{v_1, v_2, \dots, v_n\}$   
3 is the set of  $n$  nodes,  $E$  is the set of its edges,  $C = \{c_{ij} : \{v_i, v_j\} \in E\}$   
4 represents the set of edge weights, and  $w = \{w_i \geq 0 : v_i \in V\}$  is the set  
5 of node weights, the capacitated clustering problem (CCP) is to partition the

---

\* Corresponding author.

*Email addresses:* [laixiangjing@gmail.com](mailto:laixiangjing@gmail.com) (Xiangjing Lai),  
[hao@info.univ-angers.fr](mailto:hao@info.univ-angers.fr) (Jin-Kao Hao).

6 node set  $V$  into a fixed number  $p$  ( $p \leq n$  is given) of disjoint clusters (or groups)  
7 such that the sum of node weights of each cluster lies in a given interval  $[L, U]$   
8 while maximizing the sum of the edge weights whose two associated endpoints  
9 locate in the same cluster. In some related literature like [9,23], an edge weight  
10  $c_{ij} \in C$  is also called the benefit of the edge  $\{v_i, v_j\}$ , while  $L$  and  $U$  are called  
11 the lower and upper capacity limits of a cluster.

12 Formally, the CCP can be expressed as the following quadratic program with  
13 binary variables  $X_{ig}$  taking the value of 1 if node  $v_i$  is in cluster  $g$  and 0  
14 otherwise [9,23]:

$$(CCP) \quad \text{Maximize} \quad \sum_{g=1}^p \sum_{i=1}^{n-1} \sum_{j=i+1}^n c_{ij} X_{ig} X_{jg} \quad (1)$$

$$\text{Subject to} \quad \sum_{g=1}^p X_{ig} = 1, i = 1, 2, \dots, n \quad (2)$$

$$L \leq \sum_{i=1}^n w_i X_{ig} \leq U, g = 1, 2, \dots, p \quad (3)$$

$$X_{ig} \in \{0, 1\}, i = 1, 2, \dots, n; g = 1, 2, \dots, p \quad (4)$$

$$c_{ij} = 0, \forall \{v_i, v_j\} \notin E \quad (5)$$

15 where the set of constraints (2) guarantees that each node is located in exactly  
16 one cluster (or group) and the set of constraints (3) forces the sum of node  
17 weights of each cluster to be at least  $L$  and at most  $U$ . The set of constraints  
18 (5) ensures that the benefit between nodes  $v_i$  and  $v_j$  is 0 if  $\{v_i, v_j\} \notin E$ .

19 The CCP is closely related to three other clustering problems: the graph par-  
20 titioning problem (GPP) [2–4,15,30], the maximally diverse grouping prob-  
21 lem (MDGP) [5,10,14,19,28,29,33], and the handover minimization problem  
22 (HMP) [23,26]. First, the GPP is a special case of the CCP when the lower  
23 and upper capacity limits of the clusters are respectively set to 0 and  $(1 +$   
24  $\epsilon) \lceil \frac{\sum_{i=1}^n w_i}{p} \rceil$ , where  $\epsilon$  ( $\in [0, 1)$ ) is a predetermined imbalance parameter. As  
25 such, the CCP is also known as the node capacitated graph partitioning prob-  
26 lem [11–13,27] or the min-cut clustering problem [20] in the literature. Second,  
27 the MDGP is also a special case of the CCP when the given graph is a com-  
28 plete graph and the nodes have a unit weight ( $w_i = 1, i = 1, 2, \dots, n$ ) [23].  
29 Additionally, as discussed in [23,26], the HMP can be viewed as a practical  
30 application of the CCP in the context of mobile networks.

31 Given that the CCP generalizes the NP-hard MDPG, GPP, and HMP prob-  
32 lems, the CCP is at least as computationally difficult as these problems. More-  
33 over, any real-world applications that can be formulated by the MDPG, GPP,  
34 or HMP models can be cast as the CCP, such as creation of peer review groups

35 [7], parallel computing [18], assignment of students to groups [19], VLSI design  
36 [33], etc.

37 Given the NP-hard nature of the CCP and its practical importance, a large  
38 number of studies have been proposed to investigate the problem and the three  
39 related clustering problems. Below, we highlight some most recent approaches  
40 on the CCP, while refereeing the reader to two recent papers [9,23] for a  
41 comprehensive review of existing studies in the literature.

42 In 2011, Deng and Bard [9] proposed a greedy randomized adaptive search  
43 procedure with path relinking (GRASP+PR) by hybridizing a construction  
44 procedure of initial solution, a randomized variable neighborhood descent  
45 method as well as a path-relinking procedure. The reported computational  
46 results showed that the proposed GRASP+PR algorithm outperforms the ref-  
47 erence algorithms. In 2013, Morán-Mirabal et al. [26] proposed the follow-  
48 ing three heuristic algorithms for the HMP problem that is a special case of  
49 the CCP: a GRASP method (denoted by GQAP in their paper), an evolu-  
50 tionary path-relinking algorithm combined with GRASP (GevPR-HMP), and  
51 a population-based biased random-key genetic algorithm (BRKGA). Their  
52 study showed that GevPR-HMP achieved the best performance among the  
53 three proposed algorithms. In 2014, Lewis et al. [22] made a comparison be-  
54 tween the linear and nonlinear models for the CCP under the framework of  
55 exact methods, and showed that the quadratic model generally outperforms  
56 its equivalent linear alternatives.

57 Recently (2015), Martínez-Gavara et al. [23] introduced several heuristic al-  
58 gorithms for the CCP, including a new GRASP method, a tabu search (TS)  
59 method, and a hybrid algorithm combining the proposed GRASP and tabu  
60 search methods (GRASP+TS). The authors also adapted a tabu search algo-  
61 rithm with strategic oscillation (TS\_SO) originally designed for the MDGP  
62 to solve the CCP. Their study showed that the proposed GRASP+TS and  
63 TS algorithms significantly outperform their reference algorithms, including  
64 their GRASP method, Deng and Bard's GRASP and TS\_SO presented in  
65 [9], as well as the GevPR-HMP algorithm of [26]. Consequently, the TS and  
66 GRASP+TS algorithms proposed in [23] can be considered as the current best  
67 performing approaches for the CCP.

68 In this paper, we are interested in solving the general CCP problem approx-  
69 imately and propose for this purpose an effective iterated variable neighbor-  
70 hood search algorithm (IVNS). The main contributions of the present work  
71 can be highlighted as follows:

- 72 • The proposed IVNS algorithm introduces an extended variable neighbor-  
73 hood descent (EVND) method to ensure an intensified local optimization.  
74 Contrary to the standard variable neighborhood descent (VND) method

75 [24], our EVND method focuses on a more balanced exploitation between  
76 different neighborhoods, which provides the search with a reinforced diver-  
77 sification effect. Additionally, IVNS integrates two dedicated construction  
78 procedures to generate initial solutions and a randomized shake procedure  
79 to escape deep local optima (i.e., local optimum solutions which are difficult  
80 to attain and difficult to escape for a search algorithm).

81 • When it is assessed on three sets of 133 benchmark instances of the lit-  
82 erature, the proposed IVNS algorithm achieves highly competitive perfor-  
83 mances both in terms of the solution quality and computational efficiency  
84 compared to the state-of-the-art results. On the two sets of 50 standard  
85 instances, IVNS outperforms the state-of-the-art CCP algorithms in the lit-  
86 erature. Moreover, for the 83 popular benchmark instances of the third set,  
87 IVNS improves the best known results (new lower bounds) in 28 cases and  
88 matches the best known results for the 55 remaining cases.

89 The rest of the paper is organized as follows. In the next Section, our IVNS  
90 algorithm and its components are described in detail. Section 3 is dedicated  
91 to computational assessments based on the commonly used benchmarks and  
92 comparisons with the state-of-the-art algorithms in the literature. In Section  
93 4, several essential components of the proposed algorithm are investigated  
94 to shed light on how they affect the performance of the proposed algorithm.  
95 Concluding comments are summarized in the last section.

## 96 **2 Iterated Variable Neighborhood Search for the CCP**

97 Variable neighborhood search (VNS) [17,24] has been applied with success to  
98 many combinatorial optimization problems (see for instances [1,5,25,31,32]).  
99 In this work, we follow the general VNS framework and propose the iterated  
100 variable neighborhood search (IVNS) method for the CCP which integrates  
101 specially designed components to reach a suitable trade-off between inten-  
102 sification and diversification of the search process. Specifically, the proposed  
103 IVNS algorithm employs a randomized construction procedure to generate the  
104 initial solution, a new local optimization approach called the EVND method  
105 (extended variable neighborhood descent method) to discover local optima,  
106 and a shake procedure to perturb the incumbent solution. The proposed IVNS  
107 algorithm also employs a diversification stage to produce transition states be-  
108 tween high-quality local optimum solutions.

---

**Algorithm 1:** Main framework of IVNS method for CCP

---

**Input:** Instance  $I$ , parameter  $\beta_{max}$ , cutoff time  $t_{max}$ ,  $\eta$  shake strength**Output:** The best solution  $s^*$  found during the whole search process

```
1  $s \leftarrow InitialSolution(I)$  /* section 2.3 */
2  $s \leftarrow EVND(s)$  /* section 2.4.3 */
3  $s^b \leftarrow s, s^* \leftarrow s$ 
4 while  $Time() \leq t_{max}$  do
5    $\beta \leftarrow 0$ 
6   /* Intensified search: iterated local optimization with
7   Shake and EVND */
8   while  $\beta < \beta_{max} \wedge Time() \leq t_{max}$  do
9      $s \leftarrow Shake(s^b, \eta)$  /* perturb  $s^b$  before EVND improvement,
10    section 2.5 */
11     $s \leftarrow EVND(s)$  /* local improvement, section 2.4.3 */
12    if  $f(s) > f(s^*)$  then
13      |  $s^* \leftarrow s$  /* update the best solution ever found */
14    end
15    if  $f(s) > f(s^b)$  then
16      |  $s^b \leftarrow s, \beta \leftarrow 0$  /*  $s^b$  denotes the best solution obtained by
17      the current inner ‘while’ loop */
18    else
19      |  $\beta \leftarrow \beta + 1$ 
20    end
21  end
22  /* Diversification: additional Shake to escape deep local
23  optima */
24   $s^b \leftarrow Shake(s^b, \eta)$  /* an additional shake of deep local optimum
25   $s^b$  before next round of iterated local optimization */
26 end
27 return  $s^*$ 
```

---

109 *2.1 General Procedure*

110 Our IVNS algorithm (Algorithm 1) starts from an initial feasible solution  
111 that is generated by a randomized construction procedure (Section 2.3) and is  
112 improved by the EVND method (lines 1 and 2, Section 2.4.3). Then it enters  
113 a ‘while’ loop in which an iterated local optimization (the inner ‘while’ loop,  
114 lines 5 to 17) and a diversification phase (the *Shake* call, line 18) are iteratively  
115 performed until a cutoff time  $t_{max}$  is reached.

116 The inner ‘while’ loop aims to find, from a given solution (a local optimum),  
117 an improved local optimum by iterating the Shake procedure (line 7) followed  
118 by the EVND procedure (line 8). The starting solution is first shaken by mak-

119 ing  $\eta$  changes ( $\eta$  is called shake strength, see Section 2.5) which serves as the  
 120 starting point of the extended variable neighborhood descent procedure (see  
 121 Section 2.4.3). The outcome of each EVND application is used to update the  
 122 best solution ever found ( $s^*$ , lines 9-11) and the best local optimum found dur-  
 123 ing the current iterated local optimization phase ( $s^b$ , lines 12-16). The counter  
 124  $\beta$  indicates the number of consecutive local optimization (Shake+EVND) it-  
 125 erations during which  $s^b$  is not updated ( $\beta$  is reset to 0 each time an improved  
 126 local optimum  $s^b$  is discovered). The inner ‘while’ loop exits when the cutoff  
 127 time is reached (in which case the whole algorithm terminates) or when  $\beta$  at-  
 128 tains a fixed value  $\beta_{max}$  (a large  $\beta_{max}$  thus induces a more intensified search).  
 129 In the later case,  $s^b$  indicates a deep local optimum that the inner Shake call  
 130 (line 7) is not sufficient to help EVND to escape. For this reason, we apply an  
 131 additional Shake call (line 18) to modify  $s^b$  before giving it to the next round  
 132 of the inner ‘while’ loop.

133 Note that with the second Shake call (line 18), the next inner ‘while’ loop  
 134 starts the local optimization (EVND) with a doubly shaken starting solution,  
 135 which diversifies the search strongly and helps escape deep local optima. More  
 136 generally, the second Shake call may be replaced by other diversification tech-  
 137 niques like random or customized restarts. In our case, we simply adopt the  
 138 same Shake procedure used in the iterated local optimization phase. As shown  
 139 in Section 3, this technique proves to be suitable and effective for the tested  
 140 benchmarks. We also provide a study in Section 4.3 about the diversification  
 141 effect of this second Shake application.

## 142 2.2 Search Space, Evaluation Function and Solution Representation

143 For a given CCP instance that is composed of a weighted graph  $G = (V, E, C, w)$ ,  
 144 the number  $p$  of clusters, and the lower and upper limits  $L$  and  $U$  on the ca-  
 145 pacity of clusters, the search space  $\Omega$  explored by the IVNS algorithm contains  
 146 all feasible solutions, i.e., all partitions of the nodes of  $V$  into  $p$  clusters such  
 147 that the weight of each cluster lies between its lower and upper limits.

148 Formally, let  $\pi : V \rightarrow \{1, \dots, p\}$  be a partition function of the  $n$  nodes of  $V$  to  
 149  $p$  clusters. For each cluster  $g \in \{1, \dots, p\}$ , let  $\pi_g = \{v \in V : \pi(v) = g\}$  (i.e.,  
 150  $\pi_g$  is the set of nodes of cluster  $g$ ). Then the search space  $\Omega$  explored by our  
 151 IVNS algorithm is given by:

$$152 \quad \Omega = \{\pi : \forall g \in \{1, \dots, p\}, L \leq |\pi_g| \leq U, |\pi_g| = \sum_{v \in \pi_g} w(v)\}.$$

153 For any candidate partition  $s = \{\pi_1, \pi_2, \dots, \pi_p\}$  in  $\Omega$ , its quality is evaluated  
 154 by the objective function value  $f(s)$  of the CCP:

$$f(s) = \sum_{g=1}^p \sum_{v_i, v_j \in \pi_g, i < j} c_{ij} \quad (6)$$

155 Given a candidate solution  $s = \{\pi_1, \pi_2, \dots, \pi_p\}$ , IVNS employs a  $n$ -dimensional  
 156 vector  $x$  (element coordinate vector) to indicate the cluster of each node (or  
 157 element). That is, if element  $i$  belongs to cluster  $\pi_g$ , then  $x[i] = g$  ( $i \in$   
 158  $\{1, \dots, n\}$ ). IVNS additionally uses a  $p$ -dimensional vector  $WC$  (cluster weight  
 159 vector) to indicate the weight of each cluster of solution  $s$ , i.e.,  $WC[g] =$   
 160  $\sum_{v \in \pi_g} w(v)$  ( $\forall g \in \{1, \dots, p\}$ ). Moreover, to facilitate neighborhood operations  
 161 during the search process, the algorithm maintains a  $n \times p$  matrix  $\gamma$  in which  
 162 the entry  $\gamma[v][g]$  represents the sum of edge weights between the node  $v$  and  
 163 the nodes of cluster  $g$  in the candidate solution  $s$ , i.e.,  $\gamma[v][g] = \sum_{u \in \pi_g} c_{vu}$ .  
 164 Consequently, any candidate solution  $s \in \Omega$  can be conveniently indicated by  
 165 the  $x$  and  $WC$  vectors, the  $\gamma$  matrix and its objective function value  $f$ , i.e.,  $s$   
 166  $= \langle x, WC, \gamma, f \rangle$ .

### 167 2.3 Initial Solution

168 The proposed IVNS algorithm needs, for each run, an initial solution to start  
 169 its search. In this work, we devise two randomized construction procedures  
 170 for this purpose. The first procedure (Algorithm 2) operates in two stages. In  
 171 the first stage, it iteratively performs a series of insertion operations until all  
 172 clusters satisfy their lower capacity constraint. Specifically, for each insertion  
 173 operation, a node  $v$  and a cluster  $g$  are randomly chosen from the set  $AN$  of  
 174 unassigned nodes and the set  $AC$  of clusters whose lower bound constraint is  
 175 not satisfied, then the node  $v$  is assigned to cluster  $g$ . In the second stage, the  
 176 construction procedure performs again a series of insertion operations until all  
 177 nodes are assigned. Each insertion operation consists of randomly picking an  
 178 unassigned node  $v$  and a cluster  $g$  such that  $WC[g] + w(v) \leq U$ , and then  
 179 assigning  $v$  to  $g$ , where  $WC[g]$  and  $w(v)$  denote respectively the current weight  
 180 of cluster  $g$  and the weight of node  $v$ .

181 However, the preliminary experiments showed that it was often difficult to  
 182 obtain a feasible solution by the above procedure when the upper capacity  
 183 limit of clusters is very tight. As a result, we modify slightly the above proce-  
 184 dure as follows to obtain a second construction procedure. For each insertion  
 185 operation, instead of randomly picking a node from the set  $AN$  of unassigned  
 186 nodes, we always choose the node  $v$  in  $AN$  such that  $v$  has the largest weight  
 187 (break ties at random).

188 Due to the random choices for insertion operations, the construction proce-  
 189 dures are able to generate diversified initial solutions which allow the algorithm

---

**Algorithm 2:** Initial Solution Procedure

---

```
1 Function InitialSolution( )
  Input: Instance  $I$ 
  Output: A feasible initial solution (denoted by  $\langle x[1:n], WC[1:p], \gamma, f \rangle$ )
2  $AN \leftarrow \{1, 2, \dots, n\}$       /*  $AN$  is the set of available nodes */
3  $AC \leftarrow \{1, 2, \dots, p\}$     /*  $AC$  is the set of available clusters */
4 for  $g \leftarrow 1$  to  $p$  do
5   |  $WC[g] \leftarrow 0$ 
6 end
   /*  $WC[g]$  is the weight of cluster  $g$  for the current solution */
   /*  $x$  represents the coordinate vector of current solution */
7 while  $AC \neq \emptyset$  do
8   |  $v \leftarrow \text{RandomNode}(AN)$     /* randomly pick a node from  $AN$  */
9   |  $g \leftarrow \text{RandomCluster}(AC)$  /* randomly pick a cluster from  $AC$  */
10  |  $x[v] \leftarrow g$ 
11  |  $WC[g] \leftarrow WC[g] + w[v]$ 
12  |  $AN \leftarrow AN \setminus \{v\}$ 
13  | if  $WC[g] \geq L$  then
14  |   |  $AC \leftarrow AC \setminus \{g\}$ 
15  |   end
16 end
17  $AC \leftarrow \{1, 2, \dots, p\}$ 
18 while  $AN \neq \emptyset$  do
19   |  $Flag \leftarrow \text{true}$ 
20   | while  $Flag$  do
21   |   |  $v \leftarrow \text{RandomNode}(AN)$  /* randomly pick a node from  $AN$  */
22   |   |  $g \leftarrow \text{RandomCluster}(AC)$  /* randomly pick a cluster from  $AC$  */
23   |   | if  $WC[g] + w[v] \leq U$  then
24   |   |   |  $Flag \leftarrow \text{false}$ 
25   |   |   end
26   |   end
27   |    $x[v] \leftarrow g$ 
28   |    $WC[g] \leftarrow WC[g] + w[v]$ 
29   |    $AN \leftarrow AN \setminus \{v\}$ 
30 end
31 Compute  $\gamma$  and  $f$  for  $x$  /* Section 2.4.2 */
32 return  $\langle x, WC, \gamma, f \rangle$ 
```

---

190 to start each run in a different area of the search space.

#### 191 2.4 Local Optimization Method

192 Our IVNS algorithm employs the EVND method as its local optimization  
193 procedure which extends the standard variable neighborhood descent (VND)  
194 method. The detail of the EVND method is described in the following subsec-  
195 tions.

197 Our EVND procedure exploits systematically three neighborhoods, i.e., the  
 198 insertion neighborhood  $N_1$ , the swap neighborhood  $N_2$ , and the 2-1 exchange  
 199 neighborhood  $N_3$ . Note that although these three neighborhoods have been  
 200 proposed in previous studies [9,23], they have never been used together like  
 201 we do in this work.

202 The insertion neighborhood  $N_1$  is based on the *OneMove* operator. Give a so-  
 203 lution (or a partition)  $s = \{\pi_1, \pi_2, \dots, \pi_p\}$  in the search space  $\Omega$ , the *OneMove*  
 204 operator transfers a node  $v$  of  $s$  from its current cluster  $\pi_i$  to another cluster  
 205  $\pi_j$  such that  $|\pi_i| - w(v) \geq L$  and  $|\pi_j| + w(v) \leq U$ , where  $L$  and  $U$  denote  
 206 respectively the lower and upper limits of capacity of clusters,  $w(v)$  repre-  
 207 sents the weight of node  $v$ , and  $|\pi_i|$  and  $|\pi_j|$  denote respectively the weights  
 208 of the clusters  $\pi_i$  and  $\pi_j$  in  $s$ . Let  $\langle v, \pi_i, \pi_j \rangle$  designate such a move and  
 209  $s \oplus \langle v, \pi_i, \pi_j \rangle$  be the resulting neighboring solution produced by applying  
 210 the move to  $s$ . Then the neighborhood  $N_1$  of  $s$  is composed of all possible  
 211 neighboring solutions that can be obtained by applying the *OneMove* opera-  
 212 tor to  $s$ , i.e.,

$$213 \quad N_1(s) = \{s \oplus \langle v, \pi_i, \pi_j \rangle : v \in \pi_i, |\pi_i| - w(v) \geq L, |\pi_j| + w(v) \leq U, i \neq j\}$$

214 Clearly, the size of  $N_1$  is bounded by  $O(n \times p)$ .

The neighborhood  $N_2$  is defined by the *SwapMove* operator. Given two nodes  
 $v$  and  $u$  which are located in two different clusters of  $s$ , the *SwapMove*( $v, u$ )  
 operator exchanges their clusters such that the resulting neighboring solution  
 is still feasible. Thus, the neighborhood  $N_2$  of  $s$  is composed of all feasible  
 neighboring solutions that can be obtained by applying *SwapMove* to  $s$ , i.e.,

$$N_2(s) = \{s \oplus \text{SwapMove}(v, u) : v \in \pi_i, u \in \pi_j, L \leq \{|\pi_i| + w(u) - w(v), \\ |\pi_j| + w(v) - w(u)\} \leq U, i \neq j\}$$

215 The size of  $N_2$  is bounded by  $O(n^2)$  and is usually larger than that of  $N_1$ .

The neighborhood  $N_3$  is based on the 2-1 exchange operator (*Exchange*( $v, u, z$ )).  
 Given the current solution  $s = \{\pi_1, \pi_2, \dots, \pi_p\}$  and three nodes  $v, u$  and  $z$ ,  
 where  $v$  and  $u$  are located in the same cluster  $\pi_i$  and  $z$  is located in another  
 cluster  $\pi_j$ , *Exchange*( $v, u, z$ ) transfers the nodes  $v$  and  $u$  from their current  
 cluster  $\pi_i$  to the cluster  $\pi_j$ , and transfers simultaneously the node  $z$  from the  
 cluster  $\pi_j$  to the cluster  $\pi_i$  in such a way that the resulting solution is still  
 feasible. For the current solution  $s$ , the neighborhood  $N_3$  of  $s$  is composed  
 of all feasible neighboring solutions which can be obtained by applying the

$Exchange(v, u, z)$  operator to  $s$ :

$$N_3(s) = \{s \oplus Exchange(v, u, z) : v, u \in \pi_i, z \in \pi_j, L \leq \{|\pi_i| - w(u) - w(v) + w(z), |\pi_j| + w(v) + w(u) - w(z)\} \leq U, i \neq j\}$$

216 Since the  $Exchange(v, u, z)$  operator involves three nodes, the size of  $N_3$  is  
 217 bounded by  $O(n^3)$  and is usually much larger than that of  $N_2$  and  $N_1$ .

218 Additionally, it is worth noticing that these three neighborhoods (i.e.,  $N_1$ ,  $N_2$ ,  
 219 and  $N_3$ ) are functionally complementary. Actually, the  $OneMove$ ,  $SwapMove$ ,  
 220 and  $Exchange(v, u, z)$  operators transfer at a time 1, 2, and 3 nodes, respec-  
 221 tively. As a result, their combined use offers more opportunities for the local  
 222 search procedure to discover high-quality solutions.

#### 223 2.4.2 Fast Neighborhood Evaluation Technique

224 Similar to the previous studies for the MDGP [5,21,28,29,31], our EVND pro-  
 225 cedure employs an incremental evaluation technique to calculate rapidly the  
 226 move value (i.e., the change of objective value) of each candidate move. As  
 227 mentioned in Section 2.2, our procedure maintains a  $n \times p$  matrix  $\gamma$  in which  
 228 the entry  $\gamma[v][g]$  represents the sum of weights between the node  $v$  and the  
 229 nodes of cluster  $g$  for the current solution, i.e.,  $\gamma[v][g] = \sum_{u \in \pi_g} c_{vu}$ . With the  
 230 help of matrix  $\gamma$ , the evaluation function value  $f$  can be calculated as  $f(s) =$   
 231  $\frac{1}{2} \sum_{i=1}^n \gamma[i][x[i]]$  for an initial candidate solution  $s = (x[1], x[2], \dots, x[n])$ . More-  
 232 over, the matrix  $\gamma$  is frequently used in the neighborhood search operations  
 233 (see Algorithms 4 to 6).

234 Based on the current solution (or partition)  $s = \{\pi_1, \pi_2, \dots, \pi_p\}$ , if a  $OneMove$   
 235 operation  $\langle v, \pi_i, \pi_j \rangle$  is performed, the move value can be easily determined  
 236 as  $\Delta_f(\langle v, \pi_i, \pi_j \rangle) = \gamma[v][j] - \gamma[v][i]$ , and then the matrix  $\gamma$  is accordingly  
 237 updated. More specifically, the  $i$ -th and  $j$ -th columns of matrix  $\gamma$  are updated  
 238 as follows:  $\gamma[u][i] = \gamma[u][i] - c_{vu}$ ,  $\gamma[u][j] = \gamma[u][j] + c_{vu}$ ,  $\forall u \in V, u \neq v$ , where  
 239  $c_{vu}$  is the edge weight between the nodes  $v$  and  $u$ . As such, the evaluation  
 240 function value  $f$  can be rapidly updated as  $f \leftarrow f + \Delta_f$ .

241 When a  $SwapMove(v, u)$  operation is performed, its move value is calculated  
 242 as  $\Delta_f(SwapMove(v, u)) = (\gamma[v][x[u]] - \gamma[v][x[v]]) + (\gamma[u][x[v]] - \gamma[u][x[u]]) -$   
 243  $2c_{vu}$ , where  $x[v]$  and  $x[u]$  are the cluster of nodes  $v$  and  $u$  in the current  
 244 solution  $s$ . As stated in Section 2.2,  $x = (x[1], x[2], \dots, x[n])$  represents the  
 245 coordinate vector of the incumbent solution  $s$ . Since a  $SwapMove(v, u)$  oper-  
 246 ation is composed of two consecutively performed  $OneMove$  operations, i.e.,  
 247  $s \oplus SwapMove(v, u) = (s \oplus \langle v, x[v], x[u] \rangle) \oplus \langle u, x[u], x[v] \rangle$ , matrix  $\gamma$  is  
 248 consecutively updated two times according to the  $OneMove$  move.

249 When a  $Exchange(v, u, z)$  move is performed, the move value is calculated as

250  $\Delta_f(Exchange(v, u, z)) = (\gamma[v][x[z]] - \gamma[v][x[v]]) + (\gamma[u][x[z]] - \gamma[u][x[u]]) +$   
251  $(\gamma[z][x[v]] - \gamma[z][x[z]]) + 2(c_{vu} - c_{vz} - c_{uz})$ . Since a  $Exchange(v, u, z)$  move  
252 is composed of three consecutively performed  $OneMove$  moves, i.e.,  $s \oplus$   
253  $Exchange(v, u, z) = ((s \oplus \langle v, x[v], x[z] \rangle) \oplus \langle u, x[u], x[z] \rangle) \oplus \langle z, x[z], x[v] \rangle$ ,  
254 matrix  $\gamma$  is consecutively updated three times according to the  $OneMove$   
255 move.

256 Matrix  $\gamma$  is initialized at the beginning of each run of the EVND procedure  
257 with a time complexity of  $O(n^2)$ , and is updated after each move operation in  
258  $O(n)$  for any considered move operator.

### 259 2.4.3 Extended Variable Neighborhood Descent

---

**Algorithm 3:** Extended Variable Neighborhood Descent (EVND) for CCP

---

```

1 Function  $EVND(s_0)$ 
  Input: Solution  $s_0$ 
  Output: The local optimum solution  $s$ 
2  $s \leftarrow s_0$ 
3 repeat
4   repeat
5      $s \leftarrow LSN_1(s)$                                 /* Algorithm 4 */
6      $(Flag, s) \leftarrow LSN_2(s)$                        /* Algorithm 5 */
7   until  $Flag = \mathbf{false}$ 
8      $(Flag, s) \leftarrow LSN_3(s)$                        /* Algorithm 6 */
9 until  $Flag = \mathbf{false}$ 
10 return  $s$ 

```

---

260 Let  $N_k$  ( $k = 1, 2, \dots, k_{max}$ ) be a sequence of neighborhood structures (also  
261 called the neighborhood in this Section) with respect to a given optimization  
262 problem, the standard variable neighborhood descent (VND) method changes  
263 in a deterministic way the current neighborhood in order to find a high-  
264 quality local optimum solution with respect to all  $k_{max}$  neighborhoods [9,24].  
265 Specifically, the standard VND method starts with the first neighborhood  $N_1$   
266 ( $k = 1$ ) and makes a complete exploitation of the neighborhood. Then, the  
267 VND method switches orderly to the next neighborhood  $N_{k+1}$  ( $k \leftarrow k + 1$ )  
268 when the current neighborhood  $N_k$  ( $k = 1, 2, \dots, k_{max} - 1$ ) is fully explored  
269 without finding an improving solution. Moreover, the search process switches  
270 immediately to the first neighborhood  $N_1$  as soon as an improving solution  
271 is detected in the current neighborhood  $N_k$ , i.e.,  $k \leftarrow 1$ . Finally, the VND  
272 method stops if the search process reaches the last neighborhood  $N_{k_{max}}$  and  
273 no improving solution can be found in  $N_{k_{max}}$ , and the best solution found dur-  
274 ing the search process is returned as the result of the VND method. Clearly,  
275 the returned solution is a local optimum solution with respect to all  $k_{max}$   
276 neighborhoods.

---

**Algorithm 4:** Local search with  $N_1$ 

---

```
1 Function  $LSN_1(x, WC, \gamma, f)$ 
  Input:  $x[1 : n], WC[1 : p], \gamma, f$ 
  Output: The local optimum solution (denoted by  $\langle x, WC, \gamma, f \rangle$ )
  /*  $WC$  represents the weight vector of clusters */
  /*  $x$  represents the coordinate vector of current solution */
2  $Improve \leftarrow \mathbf{true}$ 
3 while  $Improve = \mathbf{true}$  do
4    $Improve \leftarrow \mathbf{false}$ 
5   for  $v \leftarrow 1$  to  $n$  do
6     for  $g \leftarrow 1$  to  $p$  do
7       if  $(x[v] \neq g) \wedge (WC[x[v]] - w[v] \geq L) \wedge (WC[g] + w[v] \leq U)$  then
8          $\Delta_f \leftarrow \gamma[v][g] - \gamma[v][x[v]]$ 
9         if  $\Delta_f > 0$  then
10           $WC[x[v]] \leftarrow WC[x[v]] - w[v]$ 
11           $WC[g] \leftarrow WC[g] + w[v]$ 
12           $x[v] \leftarrow g$ 
13           $f \leftarrow f + \Delta_f$ 
14          Update matrix  $\gamma$  /* Section 2.4.2 */
15           $Improve \leftarrow \mathbf{true}$ 
16        end
17      end
18    end
19  end
20 end
21 return  $\langle x, WC, \gamma, f \rangle$ 
```

---

277 Our EVND method described in Algorithm 3 extends the standard VND  
278 method in the sense that EVND employs a different condition to switch from  
279 the current neighborhood  $N_k$  ( $k > 1$ ) to the first neighborhood  $N_1$ . In the  
280 standard VND method, the search process switches back to the first neighbor-  
281 hood as soon as an improving solution is found in the current neighborhood  
282  $N_k$  (even if more improving solutions can be further found in  $N_k$ ). However,  
283 our EVND method switches to the first neighborhood  $N_1$  when one of the  
284 following two conditions is satisfied. First, the solution has been updated (or  
285 improved)  $m$  ( $m \geq 1$ , a parameter called ‘the depth of improvement in neigh-  
286 borhood search’) times with the current neighborhood  $N_k$ . Second, the solu-  
287 tion has been updated (improved) at least one time with the neighborhood  
288  $N_k$  and no improving solution can further be found in the neighborhood  $N_k$ .  
289 Clearly, the standard VND method is a special case of our EVND method  
290 when  $m = 1$ . Note that compared to the standard VND method, our EVND  
291 method imposes a stronger condition to move back to the first neighborhood.

292 Such an extension for the standard VND method is based on two consid-

---

**Algorithm 5:** Local search with  $N_2$ 

---

```
1 Function  $LSN_2(x, WC, \gamma, f)$ 
  Input:  $x[1:n], WC[1:p], \gamma, f, m$ 
  Output: The obtained solution (denoted by  $\langle x, WC, \gamma, f \rangle$ )
  /*  $WC$  represents the weight vector of clusters */
  /*  $x$  represents the coordinate vector of current solution */
2  $Improve \leftarrow \mathbf{true}, Flag \leftarrow \mathbf{false}, counter \leftarrow 0$ 
3 while  $Improve = \mathbf{true}$  do
4    $Improve \leftarrow \mathbf{false}$ 
5   for  $v \leftarrow 1$  to  $n - 1$  do
6     for  $u \leftarrow v + 1$  to  $n$  do
7       if  $(x[v] \neq x[u]) \wedge (L \leq WC[x[v]] + (w[u] - w[v]) \leq U) \wedge$ 
8          $(L \leq WC[x[u]] + (w[v] - w[u]) \leq U)$  then
9          $\Delta_f \leftarrow (\gamma[v][x[u]] - \gamma[v][x[v]]) + (\gamma[u][x[v]] - \gamma[u][x[u]]) - 2c_{vu}$ 
10        if  $\Delta_f > 0$  then
11           $WC[x[v]] \leftarrow WC[x[v]] + (w[u] - w[v])$ 
12           $WC[x[u]] \leftarrow WC[x[u]] + (w[v] - w[u])$ 
13           $Swap(x, v, u)$ 
14           $f \leftarrow f + \Delta_f$ 
15          Update matrix  $\gamma$  /* Section 2.4.2 */
16           $Flag \leftarrow \mathbf{true}$ 
17           $Improve \leftarrow \mathbf{true}$ 
18           $counter \leftarrow counter + 1$ 
19          if  $counter \geq m$  then
20            return  $\langle Flag, x, WC, \gamma, f \rangle$ 
21            /* Return the reached solution and stop the
22            function */
23          end
24        end
25      end
26    end
27  end
28  return  $\langle Flag, x, WC, \gamma, f \rangle$ 
```

---

293 erations. First, in the standard VND method, the first neighborhood  $N_1$  is  
294 explored more often than the other neighborhoods since we move back to  
295  $N_1$  as soon as an improving solution is discovered in the current neigh-  
296 borhood  $N_k$  ( $k > 1$ ). However, a more balanced exploitation of all the  $k$  neigh-  
297 borhoods constitutes another possibility and may help the search process  
298 to discover better solutions. Compared to the standard VND method, our  
299 EVND method promotes a more balanced exploitation of the neighborhoods  
300  $N_k$  ( $k = 2, 3, \dots, k_{max}$ ) relative to the first neighborhood  $N_1$ . Second, the solu-  
301 tions returned by the neighborhoods  $N_k$  ( $k = 2, 3, \dots, k_{max}$ ) generally have a  
302 larger distance from the local optimum solution produced most recently by the  
303 first neighborhood  $N_1$  with our EVND method than with the standard VND  
304 method. Thus, compared to the standard VND method, our EVND method  
305 creates some diversification effect during its intensified descent process with  
306 each neighborhood.

---

**Algorithm 6:** Local search with  $N_3$ 

---

```
1 Function  $LSN_3(x, WC, \gamma, f)$ 
Input:  $x[1:n], WC[1:p], \gamma, f, m$ 
Output: The obtained solution (denoted by  $\langle x, WC, \gamma, f \rangle$ )
/*  $WC$  represents the weight vector of clusters */
/*  $x$  represents the coordinate vector of current solution */
2  $Improve \leftarrow \mathbf{true}, Flag \leftarrow \mathbf{false}, counter \leftarrow 0$ 
3 while  $Improve = \mathbf{true}$  do
4    $Improve \leftarrow \mathbf{false}$ 
5   for  $v \leftarrow 1$  to  $n - 1$  do
6     for  $u \leftarrow v + 1$  to  $n$  do
7       for  $z \leftarrow 1$  to  $n$  do
8         if  $(x[v] = x[u] \wedge x[v] \neq x[z]) \wedge$ 
           $(L \leq WC[x[v]] + (w[z] - w[u] - w[v]) \leq U) \wedge$ 
           $(L \leq WC[x[z]] + (w[v] + w[u] - w[z]) \leq U)$  then
9            $\Delta_f \leftarrow (\gamma[v][x[z]] - \gamma[v][x[v]]) + (\gamma[u][x[z]] - \gamma[u][x[u]]) +$ 
             $(\gamma[z][x[v]] - \gamma[z][x[z]]) + 2(c_{vu} - c_{vz} - c_{uz})$ 
10          if  $\Delta_f > 0$  then
11             $WC[x[v]] \leftarrow WC[x[v]] + (w[z] - w[u] - w[v]),$ 
             $WC[x[z]] \leftarrow WC[x[z]] + (w[v] + w[u] - w[z])$ 
12             $swap \leftarrow x[v], x[v] \leftarrow x[z], x[u] \leftarrow x[z], x[z] \leftarrow swap$ 
13             $f \leftarrow f + \Delta_f$ 
14            Update matrix  $\gamma$  /* Section 2.4.2 */
15             $Flag \leftarrow \mathbf{ture}$ 
16             $Improve \leftarrow \mathbf{true}$ 
17             $counter \leftarrow counter + 1$ 
18            if  $counter \geq m$  then
19              return  $\langle Flag, x, WC, \gamma, f \rangle$ 
              /* Return the reached solution and stop
              the function */
20            end
21          end
22        end
23      end
24    end
25  end
26 end
27 return  $\langle Flag, x, WC, \gamma, f \rangle$ 
```

---

307 Our EVND method for the CCP exploits three complementary neighborhoods  
308 introduced in Section 2.4.1, i.e.,  $N_1$ ,  $N_2$  and  $N_3$  and is described in Algorithms  
309 3 to 6, where the variables  $x$ ,  $WC$ ,  $\gamma$  and  $f$  have the same meanings as those  
310 in Section 2.2.

311 From Algorithm 3, one can observe that our EVND procedure consists of  
312 two loops and each loop stops as long as the corresponding  $Flag$  variable  
313 receives the value  $false$ . Specifically, for the inner loop the  $Flag$  variable will  
314 get the value  $false$  when no improving neighbor exists in the neighborhood  $N_2$   
315 according to Algorithm 5. Similarly, for the outer loop the  $Flag$  variable will

316 get the value *false* when no improving neighbor exists in the neighborhood  $N_3$   
317 according to Algorithm 6. Consequently, the EVND procedure always stops  
318 when no improving neighbor exists in the neighborhoods  $N_2$  and  $N_3$  for the  
319 incumbent solution.

## 320 2.5 Shake Procedure

---

### Algorithm 7: Shake procedure

---

```

1 Function Shake( $s_0, \eta$ )
  Input: Solution  $s_0$ , strength of shake  $\eta$ 
  Output: The perturbed solution  $s$ , matrix  $\gamma$ , objective value  $f$ 
2  $s \leftarrow s_0$ 
3 for  $l \leftarrow 1$  to  $\eta$  do
4   | Randomly pick a neighboring solution  $s' \in N_1(s) \cup N_2(s)$ 
5   |  $s \leftarrow s'$ 
6 end
7 Compute  $\gamma$  and  $f$  for  $s$  /* Section 2.4.2 */
8 return  $\langle s, \gamma, f \rangle$ 

```

---

321 When our IVNS algorithm reaches a local optimum solution, we apply a *Shake*  
322 procedure to the reached solution to jump out of the local optimum trap. The  
323 *Shake* procedure used by the IVNS algorithm consists of consecutively per-  
324 forming  $\eta$  *randomly selected* feasible *OneMove* or *SwapMove* moves, where  $\eta$   
325 is a parameter called the shake strength. In other words, from the incumbent  
326 solution  $s_0$ , we construct the  $N_1(s)$  and  $N_2(s)$  neighborhoods which include all  
327 (feasible) neighboring solutions of  $s$  (see Section 2.4.1) and then pick randomly  
328 a solution  $s'$  from the union of  $N_1(s)$  and  $N_2(s)$  to replace  $s_0$ . We repeat this op-  
329 eration  $\eta$  times. It is clear that a large (respect. small)  $\eta$  value leads to a shaken  
330 solution which will be more (respect. less) distant from the input solution. In  
331 this work, the value of  $\eta$  is empirically set as  $\eta = \min\{15, \max\{5, 0.02n\}\}$ ,  
332 where  $n$  is the number of nodes in the graph. Note that it is possible to in-  
333 clude *Exchange* moves for the Shake operations. Meanwhile, for the reason of  
334 simplicity, we only apply *OneMove* and *SwapMove* moves, which proves to  
335 be sufficient for our purpose of diversification. The pseudo-code of our *Shake*  
336 procedure is given in Algorithm 7.

## 337 3 Experimental Results and Comparisons

338 In this section, we assess the performance of the proposed IVNS algorithm  
339 by showing computational results on well-known benchmark instances and by  
340 making a comparison with the state-of-the-art algorithms in the literature.

### 341 3.1 Benchmark Instances

342 Our IVNS algorithm was assessed on three sets of 133 benchmark instances  
 343 commonly used in the literature. These instances are available at [http://](http://www.opticom.es/ccp/)  
 344 [www.opticom.es/ccp/](http://www.opticom.es/ccp/), and their details are described as follows.

- 345 • **RanReal Set (40 instances):** This set was originally proposed in [14] for the  
 346 MDGP and adapted to the CCP in [23] by generating the node weights with  
 347 a uniform distribution  $U(0,10)$ . This set is composed of 20 instances with  
 348  $n = 240$ ,  $p = 12$ ,  $L = 75$ , and  $U = 125$ , and 20 instances with  $n = 480$ ,  
 349  $p = 20$ ,  $L = 100$ , and  $U = 150$ . For all instances of this set, the edge weights  
 350  $c_{ij}$  are a real number which is uniformly and randomly generated in  $(0, 100)$ .
- 351 • **DB Set (10 instances):** This set was originally proposed by Deng and Bard  
 352 [9] for the MDGP in the context of mail delivery, and adapted to the CCP  
 353 in [23] by generating the node weights with a uniform distribution  $U(0,10)$ .  
 354 These 10 instances are characterized by the following features:  $n = 82$ ,  
 355  $p = 8$ ,  $L = 25$ , and  $U = 75$ .
- 356 • **MM Set (83 instances):** These 83 synthetic instances were proposed by  
 357 Morán-Mirabal et al. [26] for the handover minimization problem and were  
 358 widely used in the literature. These instances have the following character-  
 359 istics:  $n \in \{20, 30, 40, 100, 200, 400\}$ ,  $p \in \{5, 10, 15, 25, 50\}$ , the edge weights  
 360  $c_{ij}$  are a real number, and the lower and upper capacity limits of clusters  
 361 respectively are 0 and a real number depending on each instance.

### 362 3.2 Parameter Settings and Experimental Protocol

Table 1  
Settings of parameters

Parameters	Section	Description	Values
$\beta_{max}$	2.1	strength of intensification search	30
$m$	2.4.3	depth of improvement in neighborhood search	10
$\eta$	2.5	strength of shake	$min\{15, max\{5, 0.02n\}\}$

363 In this Section, we show some basic information about our experiments, in-  
 364 cluding the parameter settings of our algorithm, the reference algorithms, the  
 365 experimental platform, and the termination criterion of algorithms.

366 First, Table 1 shows the parameter setting of our IVNS algorithm which was  
 367 achieved by a preliminary experiment. For this preliminary experiment, we  
 368 used 20 *RanReal* instances with  $n = 240$  which were also used in the sen-  
 369 sibility analysis of parameters presented in Section 4.4. The computational  
 370 results indicated that for  $m$  and  $\beta_{max}$  the default settings shown in Table 1  
 371 are suitable for the algorithm (see Section 4.4). For  $\eta$ , it involves three vari-  
 372 ables, so we manually tuned its value based on a principle that the strength of

373 shake procedure should be proportional to the size of instance but in an ap-  
374 propriate interval. The computational results on the preliminary experiment  
375 indicated that the default setting of  $\eta$  in Table 1 is able to reach an acceptable  
376 performance of the algorithm.

377 Second, according to the previous surveys [23,26], the TS [23], GRASP+TS  
378 [23], and GevPR-HMP [26] algorithms can be considered as the state-of-the-  
379 art algorithms for the CCP. Hence, in the present study we use them as the  
380 main reference algorithms for our comparative study.

381 Our IVNS algorithm was programmed in C++. To make a fair compari-  
382 son with the state-of-the-art algorithms, we also implemented faithfully the  
383 GRASP, TS, GRASP+TS algorithms of [23] which are three state-of-the-art  
384 algorithms in the literature <sup>1</sup>. For the GRASP, TS, GRASP+TS algorithms,  
385 we adopted the best parameter settings identified in the original paper [23].  
386 Moreover, all source codes were compiled using g++ compiler with the ‘-O3’  
387 flag, and the corresponding experiments were carried out on a computing plat-  
388 form with an Intel E5-2670 processor (2.80 GHz CPU and 2Gb RAM), running  
389 Linux. Following the DIMACS machine benchmark procedure, our machine re-  
390 quires respectively 0.19, 1.17, and 4.54 seconds for the graphs r300.5, r400.5,  
391 r500.5 <sup>2</sup>.

392 For the GRASP, TS, GRASP+TS, and IVNS algorithms, we used a cutoff  
393 time  $t_{max} = n$  (in seconds) as the unique stopping criterion where  $n$  is the  
394 number of nodes of the input graph.

395 Finally, for the IVNS algorithm, the initial solution was generated by the  
396 second initialization procedure for the handover minimization instances due  
397 to their tight upper bounds on the capacity of clusters, and by Algorithm 2  
398 for the remaining instances. For GRASP, TS and GRASP+TS, the handover  
399 minimization instances are not used in the experiments, since their initial  
400 solution procedures can not guarantee to generate a feasible solution for a  
401 part of them due to the tight upper bounds on the capacity of clusters.

### 402 3.3 Computational Results and Comparison on the general CCP Instances

403 The first experiment aims to assess the performance of our IVNS algorithm on  
404 the first two sets of instances by comparing its results with those of the state-  
405 of-the-art algorithms in the literature. In this experiment, all the compared  
406 algorithms (GRASP, TS, GRASP+TS and IVNS) were respectively performed

---

<sup>1</sup> The source codes of these algorithms will be available at: <http://www.info.univ-angers.fr/pub/hao/ccp.html>

<sup>2</sup> [dmclique, ftp://dimacs.rutgers.edu/pub/dsj/clique](ftp://dimacs.rutgers.edu/pub/dsj/clique)

Table 2

Comparison between the IVNS algorithm and three state-of-the-art algorithms from the literature (i.e., GRASP, TS, GRASP+TS in [25]) on the first two sets (RanReal and DB) of CCP instances in terms of the best and average objective function values over 20 independent runs. The best results among the compared algorithms are indicated in bold.

Instance	$f_{best}$				$f_{avg}$			
	GRASP	TS	GRASP+TS	IVNS	GRASP	TS	GRASP+TS	IVNS
Sparse82_01	<b>1342.17</b>	1336.82	<b>1342.17</b>	<b>1342.17</b>	1342.13	1315.21	<b>1342.17</b>	<b>1342.17</b>
Sparse82_02	<b>1306.64</b>	1303.17	<b>1306.64</b>	<b>1306.64</b>	1305.65	1281.57	1306.16	<b>1306.64</b>
Sparse82_03	<b>1353.94</b>	<b>1353.94</b>	<b>1353.94</b>	<b>1353.94</b>	1351.69	1335.89	1353.00	<b>1353.94</b>
Sparse82_04	<b>1291.22</b>	<b>1291.22</b>	<b>1291.22</b>	<b>1291.22</b>	1289.15	1276.85	1290.05	<b>1291.22</b>
Sparse82_05	<b>1352.35</b>	<b>1352.35</b>	<b>1352.35</b>	<b>1352.35</b>	<b>1352.35</b>	1328.15	<b>1352.35</b>	<b>1352.35</b>
Sparse82_06	<b>1354.61</b>	<b>1354.61</b>	<b>1354.61</b>	<b>1354.61</b>	1353.86	1329.86	<b>1354.61</b>	<b>1354.61</b>
Sparse82_07	<b>1266.94</b>	<b>1266.94</b>	<b>1266.94</b>	<b>1266.94</b>	1266.86	1227.01	1266.89	<b>1266.94</b>
Sparse82_08	<b>1393.02</b>	1391.53	<b>1393.02</b>	<b>1393.02</b>	<b>1393.02</b>	1362.54	<b>1393.02</b>	<b>1393.02</b>
Sparse82_09	<b>1294.12</b>	<b>1294.12</b>	<b>1294.12</b>	<b>1294.12</b>	1293.69	1280.97	1293.46	<b>1294.12</b>
Sparse82_10	<b>1356.98</b>	<b>1356.98</b>	<b>1356.98</b>	<b>1356.98</b>	1356.85	1330.79	1356.95	<b>1356.98</b>
RanReal240_01	192320.30	222871.98	223272.87	<b>224893.92</b>	191140.45	221547.29	222268.07	<b>224785.27</b>
RanReal240_02	185612.48	202356.36	202344.44	<b>204608.66</b>	183435.92	200582.86	200356.15	<b>204415.88</b>
RanReal240_03	179316.65	196422.35	196143.12	<b>198885.19</b>	176821.81	194348.91	194783.74	<b>198626.93</b>
RanReal240_04	197342.26	222298.86	223076.30	<b>225627.16</b>	194629.26	220521.18	221165.75	<b>225227.11</b>
RanReal240_05	175967.25	193358.53	194115.62	<b>195440.94</b>	174465.68	191234.34	192076.34	<b>195228.86</b>
RanReal240_06	192789.55	214840.96	215004.12	<b>216736.00</b>	188264.62	212626.52	213259.64	<b>216474.84</b>
RanReal240_07	191714.87	208223.05	208045.67	<b>209273.70</b>	190379.19	205808.95	206092.48	<b>209004.05</b>
RanReal240_08	185930.72	203595.13	203168.62	<b>205246.82</b>	181699.18	201102.55	201519.15	<b>204958.19</b>
RanReal240_09	189573.48	207711.19	207984.26	<b>209059.28</b>	186992.97	206540.74	206788.13	<b>208789.79</b>
RanReal240_10	176327.61	189597.87	190532.75	<b>192977.28</b>	174638.77	187534.34	188379.13	<b>192788.59</b>
RanReal240_11	184198.31	203109.91	203037.25	<b>204722.75</b>	182673.54	201113.86	201701.83	<b>204523.95</b>
RanReal240_12	181337.55	199710.82	199708.68	<b>201052.53</b>	180048.84	198365.85	198389.30	<b>200904.16</b>
RanReal240_13	180865.29	201238.18	200742.90	<b>202335.99</b>	179893.27	199590.02	198727.63	<b>202139.55</b>
RanReal240_14	194009.94	226813.94	226621.92	<b>228844.44</b>	191947.27	225362.61	225721.97	<b>228512.11</b>
RanReal240_15	173114.22	188896.07	188318.39	<b>191170.98</b>	171311.83	187410.50	187299.92	<b>190914.31</b>
RanReal240_16	182348.50	202475.44	202463.27	<b>203999.02</b>	180823.13	199999.63	200722.29	<b>203834.68</b>
RanReal240_17	181270.70	194155.13	193835.64	<b>195242.31</b>	179607.16	191978.36	191908.95	<b>195114.49</b>
RanReal240_18	174650.37	192772.21	193004.84	<b>195069.62</b>	173876.39	190428.51	190979.42	<b>194853.70</b>
RanReal240_19	179859.43	196739.04	196717.17	<b>199200.03</b>	177521.56	194916.62	195053.62	<b>199019.23</b>
RanReal240_20	191936.64	210399.94	210365.38	<b>212264.10</b>	190110.07	208970.68	208887.59	<b>212046.92</b>
RanReal480_01	463538.35	543259.55	544301.19	<b>555057.10</b>	460340.75	539074.11	539550.62	<b>554331.89</b>
RanReal480_02	446829.64	500646.59	502039.41	<b>510418.44</b>	443573.33	495273.27	497700.18	<b>509519.84</b>
RanReal480_03	434854.27	486379.91	487561.06	<b>496641.22</b>	433059.42	482624.38	483908.86	<b>495847.80</b>
RanReal480_04	455470.88	510971.67	513425.37	<b>521984.68</b>	450861.04	504054.37	507123.99	<b>520891.75</b>
RanReal480_05	415295.30	474548.74	473732.77	<b>483228.99</b>	413263.81	466932.68	469914.28	<b>482595.19</b>
RanReal480_06	461624.20	524191.64	524520.14	<b>533762.18</b>	458049.90	519002.55	520607.74	<b>532888.64</b>
RanReal480_07	461236.11	537464.01	537674.45	<b>545157.68</b>	456319.85	530513.86	532199.71	<b>544530.14</b>
RanReal480_08	460756.67	521894.07	523602.31	<b>532308.09</b>	458299.68	515438.23	518685.30	<b>531417.94</b>
RanReal480_09	466977.73	546057.32	546394.74	<b>556478.39</b>	462121.78	540459.20	541615.49	<b>555098.72</b>
RanReal480_10	447088.04	508294.13	508168.74	<b>519456.96</b>	441810.74	503161.12	504750.89	<b>518612.02</b>
RanReal480_11	451321.35	515296.61	515189.66	<b>523450.04</b>	448295.75	510410.05	509657.31	<b>522814.96</b>
RanReal480_12	434343.91	492469.23	493845.25	<b>501596.63</b>	433067.26	487442.12	488699.18	<b>500580.84</b>
RanReal480_13	467130.54	526936.22	524825.92	<b>534638.19</b>	461557.62	519201.37	521740.67	<b>533763.20</b>
RanReal480_14	428544.75	500100.04	508349.48	<b>513777.84</b>	426455.75	495859.32	499543.12	<b>512975.73</b>
RanReal480_15	446764.19	509377.07	509005.17	<b>516941.11</b>	443810.07	503159.99	503946.63	<b>516017.98</b>
RanReal480_16	465499.89	540493.75	540840.67	<b>549371.23</b>	462345.11	533502.50	535129.37	<b>548276.15</b>
RanReal480_17	460122.80	531353.71	529388.54	<b>537483.76</b>	458004.02	523459.84	524362.15	<b>536655.06</b>
RanReal480_18	456573.73	515692.20	518675.12	<b>525813.39</b>	452767.79	511193.51	512917.14	<b>524650.86</b>
RanReal480_19	454922.45	514503.74	512339.77	<b>522158.86</b>	449744.91	506858.76	508311.67	<b>521180.84</b>
RanReal480_20	443851.41	510045.22	509167.33	<b>518288.03</b>	440398.01	503995.42	505190.58	<b>517261.92</b>
#Best	10	7	10	50	2	0	4	50
<i>p-value</i>	2.54e-10	5.47e-11	2.54e-10		4.26e-12	1.54e-12	1.18e-11	

Table 3

Comparison between the IVNS algorithm and three state-of-the-art algorithms from the literature (i.e., GRASP, TS, GRASP+TS [25]) on the first two sets (RanReal and DB) of CCP instances in terms of the standard deviation and the average running time to reach its final objective value. Each instance was independently solved 20 times by each algorithm respectively.

Instance	standard deviation ( $\sigma$ )				$time_{avg}(s)$			
	GRASP	TS	GRASP+TS	IVNS	GRASP	TS	GRASP+TS	IVNS
Sparse82_01	0.16	23.24	0.00	0.00	14.01	9.31	15.15	0.13
Sparse82_02	0.93	19.99	0.97	0.00	30.82	7.53	29.27	0.82
Sparse82_03	1.24	14.90	1.63	0.00	41.60	2.38	38.78	0.21
Sparse82_04	1.32	12.97	1.04	0.00	30.60	6.36	36.99	4.27
Sparse82_05	0.00	30.33	0.00	0.00	4.97	8.02	5.11	0.07
Sparse82_06	1.39	25.63	0.00	0.00	24.43	4.03	26.04	0.08
Sparse82_07	0.18	22.46	0.10	0.00	32.74	4.75	28.10	0.39
Sparse82_08	0.00	29.24	0.00	0.00	1.00	9.31	0.74	0.03
Sparse82_09	0.40	13.01	0.22	0.00	28.54	5.67	16.31	0.58
Sparse82_10	0.16	24.48	0.07	0.00	29.41	2.08	37.81	0.59
RanReal240_01	864.05	1098.76	601.21	97.88	166.10	15.24	132.87	147.28
RanReal240_02	936.74	1140.64	1886.92	102.77	128.76	17.81	132.97	160.27
RanReal240_03	786.12	1192.60	875.04	170.83	123.40	7.19	125.43	146.88
RanReal240_04	1018.21	1302.30	1060.61	237.09	122.63	13.49	146.27	162.63
RanReal240_05	646.56	1255.73	894.36	91.62	113.10	8.38	129.66	161.77
RanReal240_06	1321.33	1251.75	959.32	169.61	128.25	15.04	139.80	165.88
RanReal240_07	592.59	1696.59	1324.09	120.82	114.48	27.88	143.93	120.41
RanReal240_08	1271.82	1184.11	1096.61	139.71	137.49	19.52	128.97	174.29
RanReal240_09	1127.57	1073.39	873.02	148.77	135.58	27.24	158.97	160.17
RanReal240_10	659.31	1160.27	1080.52	154.18	107.53	7.40	127.40	170.78
RanReal240_11	855.28	1127.48	1041.35	96.57	115.17	36.41	162.40	148.56
RanReal240_12	624.13	1347.08	961.73	126.79	128.65	34.82	151.19	169.33
RanReal240_13	531.24	1095.26	1191.67	155.41	102.67	38.44	131.83	150.75
RanReal240_14	1185.33	1086.76	672.46	170.71	112.17	10.11	124.59	141.27
RanReal240_15	780.67	1026.16	700.91	174.08	120.50	8.78	126.35	144.95
RanReal240_16	645.97	1341.25	1033.99	131.32	97.79	19.38	141.58	147.86
RanReal240_17	679.79	1280.03	936.77	109.08	120.85	25.01	142.44	146.03
RanReal240_18	458.00	1155.66	1050.83	126.17	141.20	7.55	130.71	163.49
RanReal240_19	924.99	1173.30	1357.67	109.79	113.09	8.71	125.27	176.31
RanReal240_20	829.60	908.52	816.33	130.92	99.00	24.49	146.64	149.66
RanReal480_01	1767.54	2915.75	2305.94	415.49	276.78	87.81	270.05	369.26
RanReal480_02	1541.69	3027.29	1982.60	569.71	212.88	50.36	276.25	416.40
RanReal480_03	909.49	2529.01	1999.88	426.86	281.57	55.11	274.68	415.80
RanReal480_04	1426.35	4323.08	2867.38	648.97	230.07	53.23	275.02	380.20
RanReal480_05	1094.52	3383.90	1886.34	497.25	310.87	33.56	271.51	338.96
RanReal480_06	1573.22	3178.92	2393.40	555.89	205.83	58.20	291.32	338.25
RanReal480_07	1926.89	2864.48	2451.06	413.13	198.46	63.44	291.77	388.74
RanReal480_08	1404.82	4291.62	2058.61	555.42	194.59	77.79	291.08	373.48
RanReal480_09	1994.95	3774.05	2163.84	514.64	231.09	78.00	282.38	388.08
RanReal480_10	1667.97	2799.90	2715.16	589.69	296.32	49.75	268.99	403.46
RanReal480_11	1488.69	2771.21	2220.12	402.37	243.62	98.13	277.81	382.52
RanReal480_12	1078.08	3170.40	2713.09	540.16	292.27	80.68	296.84	386.04
RanReal480_13	2379.98	3611.97	2431.74	423.89	264.67	66.61	289.75	420.59
RanReal480_14	1025.07	2482.93	2984.86	408.22	275.91	41.00	276.64	401.09
RanReal480_15	1358.30	3001.21	3149.15	408.07	241.74	86.60	296.67	375.96
RanReal480_16	1331.41	4488.81	2636.34	590.34	217.53	56.34	279.50	366.12
RanReal480_17	1306.58	3384.98	2538.33	402.38	274.19	56.09	286.52	389.54
RanReal480_18	1343.73	4530.39	2388.17	494.86	248.92	80.52	307.61	379.04
RanReal480_19	2115.36	2949.64	2357.29	550.77	248.76	49.19	282.87	405.61
RanReal480_20	1146.60	3505.30	2635.41	530.58	255.89	91.93	288.15	398.10
#Best	2	0	4	50				
<i>p-value</i>	4.26e-12	1.54e-12	1.18e-11					

407 20 times on each instance, based on the experimental protocol of Section 3.2.  
408 The computational results are summarized in Tables 2 and 3.

409 In Table 2, the first column identifies the instances, columns 2–4 show respec-  
410 tively the best objective values ( $f_{best}$ ) obtained by the three reference algo-  
411 rithms (GRASP, TS, GRASP+TS), and column 5 reports the best objective  
412 values of our IVNS algorithm. Columns 6–9 show respectively the average ob-  
413 jective values for the four compared algorithms ( $f_{avg}$ ). The best results among  
414 the algorithms in terms of the best and average objective values are indicated  
415 in bold. In Table 3, columns 2–5 show the standard deviation ( $\sigma$ ) of the objec-  
416 tive values obtained over 20 runs for the compared algorithms, respectively,  
417 and columns 6–9 give the average running times (in seconds) of the algo-  
418 rithms to reach their respective objective values ( $time_{avg}$ ). The row *#Best* of  
419 the tables indicates the number of instances for which the corresponding algo-  
420 rithm produces the best results among the compared algorithms. In addition,  
421 to verify whether there exists a significant difference between the reference  
422 algorithms and our IVNS algorithm on the best and average objective val-  
423 ues, as well as the standard deviation of objective values, the  $p$ -values from  
424 the non-parametric Friedman test are reported in the last row of the tables.  
425 Notice that a  $p$ -value smaller than 0.05 means that there exists a significant  
426 difference between two sets of results compared.

427 One observes from Tables 2 and 3 that the proposed IVNS algorithm outper-  
428 forms the reference algorithms. First, IVNS obtained the best result on all  
429 50 instances in terms of the best objective value, whereas the GRASP, TS,  
430 GRASP+TS algorithms produced respectively the best result on 10, 7 and  
431 10 instances. Second, when comparing the average objective values, it can be  
432 found that the IVNS algorithm yielded the best result on all instances, whereas  
433 the GRASP, TS, and GRASP+TS algorithms respectively obtained the best  
434 results on only 2, 0, 4 instances. In addition, the small  $p$ -values ( $< 0.05$ ) con-  
435 firm the significant differences between the results of IVNS and those of the  
436 compared reference algorithms.

437 Finally, compared to the reference algorithms, the IVNS algorithm produced  
438 the smallest standard deviation ( $\sigma$ ) on all tested instances, indicating that  
439 IVNS is the most robust algorithm among the compared algorithms, which is  
440 also confirmed by the associated small  $p$ -values.

### 441 3.4 Computational Results and Comparison on the Handover Minimization 442 Instances

443 The second experiment aims to assess the performance of the IVNS algorithm  
444 on the set of 83 handover minimization instances with  $n \leq 400$ , where for each

Table 4

Comparison between the IVNS algorithm and the three reference algorithms in [26] on the set of handover minimization instances. Each instance was independently solved 20 times by the IVNS algorithm, and the current best results are indicated in bold. The results are given in the form of minimization to make a direct comparison with the results from the literature.

Instance	BKS	GevPR-HMP	GQAP	BRKGA	IVNS				
					$f_{best}$	$f_{avg}$	$f_{worst}$	$\sigma$	$time_{avg}(s)$
100_15_270001	<b>19000</b>	19174	<b>19000</b>	<b>19000</b>	<b>19000</b>	19000.00	19000	0.00	1.01
100_15_270002	<b>22686</b>	<b>22686</b>	<b>22686</b>	23288	<b>22686</b>	22686.00	22686	0.00	0.71
100_15_270003	<b>14558</b>	<b>14558</b>	<b>14558</b>	14616	<b>14558</b>	14558.00	14558	0.00	0.09
100_15_270004	<b>19700</b>	19762	<b>19700</b>	19882	<b>19700</b>	19700.00	19700	0.00	0.15
100_15_270005	<b>22746</b>	22892	<b>22746</b>	23092	<b>22746</b>	22746.00	22746	0.00	0.67
100_25_270001	<b>36412</b>	<b>36412</b>	36448	36752	<b>36412</b>	36412.00	36412	0.00	2.42
100_25_270002	<b>38608</b>	39144	<b>38608</b>	39256	<b>38608</b>	38608.00	38608	0.00	1.04
100_25_270003	<b>32686</b>	32966	<b>32686</b>	32708	<b>32686</b>	32686.00	32686	0.00	1.82
100_25_270004	<b>35322</b>	35678	<b>35322</b>	35954	<b>35322</b>	35322.00	35322	0.00	0.26
100_25_270005	36878	36906	36878	37100	<b>36690</b>	36690.00	36690	0.00	0.21
100_50_270001	<b>60922</b>	<b>60922</b>	61172	61554	<b>60922</b>	60922.00	60922	0.00	2.58
100_50_270002	<b>62022</b>	62046	<b>62022</b>	62524	<b>62022</b>	62022.00	62022	0.00	0.53
100_50_270003	<b>54596</b>	54618	<b>54596</b>	55192	<b>54596</b>	54596.00	54596	0.00	4.07
100_50_270004	<b>57894</b>	<b>57894</b>	<b>57894</b>	58208	<b>57894</b>	57894.00	57894	0.00	1.18
100_50_270005	61088	61088	61318	62784	<b>61080</b>	61082.80	61090	4.31	46.02
200_15_270001	<b>81558</b>	<b>81558</b>	82834	<b>81558</b>	<b>81558</b>	81558.00	81558	0.00	12.53
200_15_270002	89810	89810	90620	90506	<b>89492</b>	90502.80	91172	546.53	48.73
200_15_270003	<b>79232</b>	<b>79232</b>	80980	79548	<b>79232</b>	79277.60	80144	198.77	13.39
200_15_270004	<b>78324</b>	<b>78324</b>	80538	80026	<b>78324</b>	78485.50	79726	375.08	52.95
200_15_270005	95998	95998	98826	98830	<b>95680</b>	96137.10	96986	622.92	21.85
200_25_270001	<b>133168</b>	<b>133168</b>	138454	140492	<b>133168</b>	133168.00	133168	0.00	51.17
200_25_270002	136038	136038	140066	140690	<b>133778</b>	133859.80	133926	47.19	68.02
200_25_270003	139438	139438	144120	143724	<b>136782</b>	136795.50	136812	14.92	67.84
200_25_270004	128554	128554	134054	131786	<b>128246</b>	128246.00	128246	0.00	53.92
200_25_270005	148402	148402	154260	152934	<b>147844</b>	147844.00	147844	0.00	10.20
200_50_270001	219672	221550	223096	223098	<b>215388</b>	215531.20	215572	64.28	67.70
200_50_270002	216444	218254	219910	219834	<b>212798</b>	212864.60	212912	36.18	92.01
200_50_270003	221348	221500	222404	221110	<b>214364</b>	214413.90	214426	15.03	61.89
200_50_270004	211832	212044	212544	213170	<b>206476</b>	206509.80	206590	29.01	61.66
200_50_270005	231890	231890	236136	237156	<b>229918</b>	230050.70	230082	22.82	76.55
400_15_270001	370314	372694	456158	375650	<b>369048</b>	372055.40	385786	4442.20	187.66
400_15_270002	370274	370274	460232	383096	<b>365878</b>	369275.90	378508	4126.82	228.23
400_15_270003	358684	358684	448830	366314	<b>352588</b>	356988.80	365886	4140.92	135.41
400_15_270004	334430	334430	406834	346282	<b>331888</b>	339169.90	350388	6361.04	214.63
400_15_270005	361904	361904	457274	377094	<b>360422</b>	363890.10	383154	4970.80	200.30
400_25_270001	568830	570852	663908	579130	<b>545118</b>	546936.70	549318	1026.66	208.04
400_25_270002	543182	544568	658440	554840	<b>528470</b>	529087.80	530118	444.68	201.85
400_25_270003	548000	548000	667982	553162	<b>524678</b>	526220.60	530438	1695.07	215.17
400_25_270004	501750	501750	607672	516416	<b>481568</b>	482070.00	484566	722.65	174.58
400_25_270005	556044	556044	679848	585070	<b>548100</b>	549839.10	557482	2608.26	240.02
400_50_270001	851412	851412	951882	879438	<b>824766</b>	825581.20	826202	402.47	241.94
400_50_270002	845496	845496	949562	874226	<b>823094</b>	824239.00	825442	591.82	216.54
400_50_270003	819242	819242	919140	843242	<b>801586</b>	802672.90	804310	776.71	266.22
400_50_270004	774564	774564	878912	806690	<b>760602</b>	761370.40	763076	553.34	275.49
400_50_270005	854726	854726	940358	882060	<b>828384</b>	829335.50	830116	421.66	238.48
#Improve	0	0	0	0	28				
#Match	17	9	11	2	17				
#Total	45	45	45	45	45				
<i>p-value</i>	1.21e-7	1.97e-9	5.51e-9	5.47e-11					

445 instance the IVNS algorithm was independently run 20 times. The computa-  
446 tional results are summarized in Table 4 for the large instances with  $n \geq 100$ .  
447 For very small instances with  $n \leq 40$ , the computational results are reported  
448 in Appendix (Table 10) since they are very easy to be solved by the IVNS  
449 algorithm (see Appendix for the details). Notice that in the present section  
450 all results are given in the form of minimization to make a direct compar-  
451 ison between the results of the IVNS algorithm and those reported in the

452 literature, and that the results of the maximization form can be converted to  
 453 the minimization form as follows:  $f_{min} = 2(\sum_{i<j} c_{ij} - f_{max})$ , where  $f_{min}$  and  
 454  $f_{max}$  respectively correspond to the results of minimization and maximization  
 455 forms.

456 Columns 1 and 2 of Table 4 respectively give the instance name and the  
 457 best known solution (BKS) published in the literature. Columns 3–5 show the  
 458 best results of three reference algorithms in [26]: a GRASP method (GQAP),  
 459 a GRASP embedded within a population-based evolutionary path-relinking  
 460 algorithm (GevPR-HMP), and a population-based biased random-key genetic  
 461 algorithm (BRKGA). The results of these reference algorithms were directly  
 462 extracted from [26], which correspond to the best outcomes ( $f_{best}$ ) yielded by  
 463 5 runs with a cutoff time of 24 hours based on a cluster running Intel X5650  
 464 processors at 2.67 GHz or a cluster running Intel Xeon E5530 processors at  
 465 2.4 GHz [26]. It is worth noting that the cutoff time of the three reference  
 466 algorithms is much higher than ours (24 hours vs.  $n \leq 400$  seconds). Columns  
 467 6–10 show the results of our IVNS algorithm, including the best objective value  
 468 ( $f_{best}$ ) over 20 runs, the average objective value ( $f_{avg}$ ), the worst objective value  
 469 ( $f_{worst}$ ), the standard deviation of objective value ( $\sigma$ ), and the average running  
 470 time in seconds to reach its final objective value ( $time_{avg}$ ). The rows *Improve*,  
 471 *Match* denote the number of instances for which the associated algorithm  
 472 improved or matched the best known results in the literature, and row *Total*  
 473 shows the total number of instances. Note that the current best results are  
 474 indicated in bold, and other symbols are the same as those in Table 2. Besides,  
 475 it should be mentioned that this section focuses on the best results produced by  
 476 the compared algorithms, since the compared algorithms were run on different  
 477 computers and the cut-off times of the reference algorithms are much longer  
 478 than that of our IVNS algorithm.

479 Table 4 clearly discloses that the proposed IVNS algorithm outperforms the  
 480 three reference algorithms designed for the handover minimization problem.  
 481 First, the IVNS algorithm improved the best known results for 28 out of 45  
 482 instances with  $n \geq 100$ , while matching the best known results for the remain-  
 483 ing instances. Second, compared to any of the three reference algorithms, our  
 484 IVNS algorithm obtained the better or equal objective values for all instances,  
 485 even if IVNS uses much shorter cutoff times than that of the reference algo-  
 486 rithms ( $n \leq 400$  seconds vs. 24 hours). Third, even the worst objective value  
 487 produced by the IVNS algorithm is better than the best known result reported  
 488 in the literature for instances with  $n = 400$ , and the average computing time  
 489  $time_{avg}$  is smaller than 300 for each instance. Finally, one observes that all  
 490 *p-values* are smaller than 0.05, implying there exists a significant difference  
 491 between the results of the IVNS algorithm and those yielded by the reference  
 492 algorithms. In summary, these outcomes indicate that the proposed IVNS al-  
 493 gorithm is highly efficient for solving the handover minimization instances  
 494 compared to the state-of-the-art algorithms in the literature [26].

## 495 4 Analysis and Discussions

496 We now turn our attention to analyze some essential aspects of the proposed  
497 IVNS algorithm, including the local optimization procedure (i.e., the EVND  
498 method), the influence of the diversification stage on the performance of IVNS  
499 algorithm, and a sensitivity analysis of the key parameters. In this section,  
500 all experiments were carried out based on the set of *RanReal* instances (20  
501 instances with  $n = 240$  and 20 instances with  $n = 480$ ).

### 502 4.1 Comparison Between the Standard and Extended VND Methods

503 The IVNS algorithm employs the extended VND method (EVND) as its local  
504 optimization procedure. Since the EVND method is an extension of the stan-  
505 dard VND method, we carried out an experiment to compare both methods.  
506 In this experiment, both EVND and VND were respectively run 100 times  
507 on each instance. Specifically, for each run, both methods were performed  
508 with the same initial solution generated by the first construction procedure  
509 presented in Section 2.3.

510 The computational results of this experiment are summarized in Table 5, in-  
511 cluding the average objective function value ( $f_{avg}$ ) and the average running  
512 time ( $time_{avg}$ ). In addition, the rows *Better*, *Equal* and *Worse* of the table de-  
513 note the number of instances for which the corresponding algorithm obtained  
514 a better, equal, and worse average objective value compared to another one.  
515 The  $p$ -values from the non-parametric Friedman test are given in the last row  
516 of the table.

517 It can be observed from Table 5 that in terms of the average objective value,  
518 the EVND method achieves a better result than the standard VND method  
519 for 36 out of 40 instances, whereas both methods consumed a similar compu-  
520 tational time for most instances. These outcomes demonstrate the interest of  
521 the EVND method compared to the standard VND method.

### 522 4.2 Importance of 2-1 Exchange Neighborhood $N_3$

523 The EVND method employs three complementary neighborhoods i.e.,  $N_1$ ,  $N_2$   
524 and the 2-1 exchange neighborhood  $N_3$ . While  $N_1$  and  $N_2$  are very popular  
525 and their effectiveness has been shown on a number of the clustering problems  
526 in the literature [5,6,9,28,29,31],  $N_3$  is not well studied and thus less under-  
527 stood. In this section, we assess the influence of  $N_3$  on the performance of the  
528 IVNS algorithm. The computational experiment was carried out as follows.

Table 5

Comparison between the standard VND method and the extended VND (EVND) method on the set of 40 representative instances. Each instance was independently solved 100 times by both algorithms respectively, and better results in the average objective value ( $f_{avg}$ ) between the compared algorithms are indicated in bold.

Instance	$f_{avg}$		$time_{avg}$	
	VND	EVND	VND	EVND
RanReal240_01	220221.45	<b>221035.18</b>	0.12	0.12
RanReal240_02	199165.17	<b>199461.37</b>	0.09	0.10
RanReal240_03	193878.39	<b>194087.07</b>	0.09	0.10
RanReal240_04	219242.90	<b>220280.46</b>	0.08	0.11
RanReal240_05	190443.88	<b>190569.72</b>	0.08	0.10
RanReal240_06	211538.07	<b>212198.32</b>	0.09	0.10
RanReal240_07	203850.69	<b>204429.68</b>	0.10	0.13
RanReal240_08	200600.21	<b>200710.54</b>	0.13	0.12
RanReal240_09	204291.07	<b>204961.02</b>	0.08	0.10
RanReal240_10	186995.53	<b>187202.99</b>	0.09	0.11
RanReal240_11	199062.90	<b>199574.51</b>	0.09	0.11
RanReal240_12	196535.07	<b>196618.48</b>	0.08	0.11
RanReal240_13	197326.13	<b>197585.12</b>	0.10	0.11
RanReal240_14	224438.53	<b>224784.93</b>	0.12	0.12
RanReal240_15	185489.25	<b>186227.47</b>	0.08	0.10
RanReal240_16	198794.30	<b>199277.26</b>	0.10	0.11
RanReal240_17	189651.07	<b>190188.43</b>	0.07	0.10
RanReal240_18	189290.28	<b>189691.65</b>	0.10	0.10
RanReal240_19	193267.47	<b>194274.68</b>	0.08	0.10
RanReal240_20	207193.59	<b>207692.77</b>	0.08	0.10
RanReal480_01	541860.57	<b>545446.40</b>	0.88	0.87
RanReal480_02	497118.68	<b>498148.99</b>	1.01	0.95
RanReal480_03	<b>483285.49</b>	482184.26	0.99	0.99
RanReal480_04	507951.82	<b>509674.81</b>	0.76	0.90
RanReal480_05	<b>469493.30</b>	469172.22	0.86	0.91
RanReal480_06	516732.69	<b>518796.77</b>	0.96	0.95
RanReal480_07	528136.56	<b>533541.23</b>	0.99	0.93
RanReal480_08	516413.17	<b>518435.89</b>	0.81	0.83
RanReal480_09	543150.66	<b>546057.68</b>	0.88	0.98
RanReal480_10	507686.31	<b>508665.48</b>	0.77	0.86
RanReal480_11	511503.09	<b>512682.60</b>	0.96	0.94
RanReal480_12	487411.85	<b>488100.45</b>	1.04	1.08
RanReal480_13	517853.71	<b>521632.82</b>	0.87	0.88
RanReal480_14	<b>500399.35</b>	500139.40	1.01	0.94
RanReal480_15	501547.03	<b>503165.49</b>	0.88	0.83
RanReal480_16	536611.41	<b>537921.48</b>	1.00	0.86
RanReal480_17	526315.11	<b>526884.43</b>	0.92	0.95
RanReal480_18	508572.09	<b>511441.64</b>	0.88	0.85
RanReal480_19	<b>509748.10</b>	509208.68	0.70	0.84
RanReal480_20	502534.92	<b>504453.71</b>	0.93	0.87
#Better	4	36		
#Equal	0	0		
#Worse	36	4		
$p$ -value		4.20e-7		1.96e-2

529 We ran our IVNS and IVNS<sup>-</sup> methods 20 times to solve each instance, where  
530 IVNS<sup>-</sup> is a variant of IVNS in which  $N_3$  (corresponding to subroutine  $LSN_3$   
531 of Algorithm 6) is disabled while keeping the other algorithmic ingredients  
532 unchanged. The experimental results are summarized in Table 6, including  
533 the average objective value  $f_{avg}$ , the standard deviation of objective value ( $\sigma$ ),  
534 and the average running time to reach its final objective value ( $time_{avg}$ ), and  
535 other symbols are the same as those in the previous tables.

536 Table 6 shows that without  $N_3$ , the performance of IVNS deteriorates for all  
537 instances in terms of average objective value. Moreover, the average computing  
538 times indicate that  $N_3$  helps the IVNS algorithm to continue its search for a

Table 6

Comparison between the IVNS method and its a variant (IVNS<sup>-</sup>) in which the neighborhood  $N_3$  is disabled on the set of 40 representative instances. Each instance is respectively solved 20 times by both algorithms, and better results in the average objective value between two algorithms are indicated in bold.

Instance	$f_{avg}$		$\sigma$		$time_{avg}$	
	IVNS <sup>-</sup>	IVNS	IVNS <sup>-</sup>	IVNS	IVNS <sup>-</sup>	IVNS
RanReal240_01	223891.05	<b>224785.27</b>	179.47	97.88	114.90	147.28
RanReal240_02	203789.58	<b>204415.88</b>	160.64	102.77	120.46	160.27
RanReal240_03	198236.57	<b>198626.93</b>	96.21	170.83	145.08	146.88
RanReal240_04	224617.33	<b>225227.11</b>	86.58	237.09	149.30	162.63
RanReal240_05	195181.62	<b>195228.86</b>	82.96	91.62	149.44	161.77
RanReal240_06	215491.95	<b>216474.84</b>	126.77	169.61	109.82	165.88
RanReal240_07	208889.10	<b>209004.05</b>	74.03	120.82	147.34	120.41
RanReal240_08	204207.83	<b>204958.19</b>	230.94	139.71	158.62	174.29
RanReal240_09	208562.04	<b>208789.79</b>	50.70	148.77	119.07	160.17
RanReal240_10	192253.79	<b>192788.59</b>	100.09	154.18	147.11	170.78
RanReal240_11	203776.92	<b>204523.95</b>	109.20	96.57	149.70	148.56
RanReal240_12	200251.35	<b>200904.16</b>	145.50	126.79	101.56	169.33
RanReal240_13	201581.76	<b>202139.55</b>	143.75	155.41	123.42	150.75
RanReal240_14	228332.63	<b>228512.11</b>	228.63	170.71	108.74	141.27
RanReal240_15	190130.92	<b>190914.31</b>	190.24	174.08	137.21	144.95
RanReal240_16	203072.49	<b>203834.68</b>	197.23	131.32	110.16	147.86
RanReal240_17	194564.42	<b>195114.49</b>	92.24	109.08	113.22	146.03
RanReal240_18	194370.83	<b>194853.70</b>	121.64	126.17	107.11	163.49
RanReal240_19	198496.90	<b>199019.23</b>	144.56	109.79	111.23	176.31
RanReal240_20	211418.56	<b>212046.92</b>	130.93	130.92	106.77	149.66
RanReal480_01	550160.70	<b>554331.89</b>	777.24	415.49	304.77	369.26
RanReal480_02	507893.39	<b>509519.84</b>	457.07	569.71	286.47	416.40
RanReal480_03	493226.43	<b>495847.80</b>	584.05	426.86	307.52	415.80
RanReal480_04	518088.82	<b>520891.75</b>	462.43	648.97	303.57	380.20
RanReal480_05	481778.58	<b>482595.19</b>	436.13	497.25	310.42	338.96
RanReal480_06	529846.53	<b>532888.64</b>	537.12	555.89	240.48	338.25
RanReal480_07	542575.23	<b>544530.14</b>	675.84	413.13	310.92	388.74
RanReal480_08	529264.25	<b>531417.94</b>	464.02	555.42	265.15	373.48
RanReal480_09	551329.66	<b>555098.72</b>	547.85	514.64	246.68	388.08
RanReal480_10	516929.26	<b>518612.02</b>	466.20	589.69	282.97	403.46
RanReal480_11	520357.01	<b>522814.96</b>	424.04	402.37	318.02	382.52
RanReal480_12	498472.55	<b>500580.84</b>	410.95	540.16	258.80	386.04
RanReal480_13	530678.39	<b>533763.20</b>	628.41	423.89	261.10	420.59
RanReal480_14	509895.03	<b>512975.73</b>	685.12	408.22	251.09	401.09
RanReal480_15	513947.87	<b>516017.98</b>	645.76	408.07	255.18	375.96
RanReal480_16	544921.69	<b>548276.15</b>	933.12	590.34	332.77	366.12
RanReal480_17	533634.44	<b>536655.06</b>	498.31	402.38	277.45	389.54
RanReal480_18	521497.91	<b>524650.86</b>	775.43	494.86	288.94	379.04
RanReal480_19	519184.62	<b>521180.84</b>	716.91	550.77	230.32	405.61
RanReal480_20	514755.82	<b>517261.92</b>	500.77	530.58	286.77	398.10
#Better	0	40				
#Equal	0	0				
#Worse	40	0				
<i>p-value</i>		2.54e-10				

539 longer time and thus to attain better solutions. This experiment demonstrates  
540 the usefulness of the 2-1 exchange neighborhood for the IVNS algorithm.

#### 541 4.3 Importance of the Diversification Mechanism

542 The IVNS algorithm performs an intensified search stage with the iterated  
543 local optimization (lines 5–17 of Algorithm 1) and a diversified stage with  
544 the Shake procedure (line 18 of Algorithm 1). The diversified stage aims at  
545 producing transition states between two high-quality local optima, since these  
546 transition states are usually necessary to help the search process to move from

Table 7

Comparative results of the IVNS method with and without its diversified stage (IVNS-D), on the set of 40 representative instances. Each instance was independently solved 20 times by both algorithms respectively, and better results in terms of the average objective value between two algorithms are indicated in bold.

Instance	$f_{avg}$		$\sigma$		$time_{avg}$	
	IVNS-D	IVNS	IVNS-D	IVNS	IVNS-D	IVNS
RanReal240_01	223986.99	<b>224785.27</b>	272.57	97.88	69.37	147.28
RanReal240_02	203614.45	<b>204415.88</b>	299.81	102.77	113.36	160.27
RanReal240_03	197731.77	<b>198626.93</b>	418.98	170.83	100.27	146.88
RanReal240_04	224424.68	<b>225227.11</b>	459.32	237.09	93.67	162.63
RanReal240_05	194298.12	<b>195228.86</b>	474.64	91.62	113.46	161.77
RanReal240_06	215609.74	<b>216474.84</b>	318.55	169.61	90.56	165.88
RanReal240_07	208341.50	<b>209004.05</b>	378.79	120.82	122.25	120.41
RanReal240_08	204211.41	<b>204958.19</b>	219.57	139.71	93.92	174.29
RanReal240_09	208092.99	<b>208789.79</b>	286.18	148.77	109.41	160.17
RanReal240_10	191828.97	<b>192788.59</b>	482.02	154.18	133.85	170.78
RanReal240_11	203921.95	<b>204523.95</b>	329.82	96.57	75.89	148.56
RanReal240_12	199971.56	<b>200904.16</b>	300.91	126.79	105.04	169.33
RanReal240_13	201224.78	<b>202139.55</b>	467.84	155.41	87.82	150.75
RanReal240_14	227825.89	<b>228512.11</b>	356.96	170.71	103.34	141.27
RanReal240_15	189814.49	<b>190914.31</b>	446.62	174.08	81.73	144.95
RanReal240_16	202951.20	<b>203834.68</b>	412.54	131.32	87.63	147.86
RanReal240_17	194328.55	<b>195114.49</b>	265.17	109.08	165.12	146.03
RanReal240_18	193915.24	<b>194853.70</b>	286.06	126.17	104.30	163.49
RanReal240_19	197900.00	<b>199019.23</b>	461.25	109.79	101.25	176.31
RanReal240_20	211284.31	<b>212046.92</b>	280.17	130.92	96.18	149.66
RanReal480_01	552979.67	<b>554331.89</b>	560.30	415.49	375.66	369.26
RanReal480_02	507974.23	<b>509519.84</b>	816.89	569.71	337.88	416.40
RanReal480_03	494000.83	<b>495847.80</b>	961.92	426.86	372.23	415.80
RanReal480_04	519475.06	<b>520891.75</b>	843.47	648.97	367.62	380.20
RanReal480_05	481098.07	<b>482595.19</b>	832.34	497.25	315.49	338.96
RanReal480_06	531667.60	<b>532888.64</b>	649.14	555.89	360.97	338.25
RanReal480_07	543160.19	<b>544530.14</b>	666.03	413.13	345.01	388.74
RanReal480_08	530127.30	<b>531417.94</b>	626.52	555.42	369.13	373.48
RanReal480_09	553832.78	<b>555098.72</b>	703.40	514.64	383.55	388.08
RanReal480_10	516798.04	<b>518612.02</b>	653.19	589.69	327.60	403.46
RanReal480_11	520835.32	<b>522814.96</b>	952.45	402.37	359.04	382.52
RanReal480_12	499078.98	<b>500580.84</b>	687.26	540.16	334.42	386.04
RanReal480_13	532308.08	<b>533763.20</b>	792.43	423.89	390.62	420.59
RanReal480_14	511808.09	<b>512975.73</b>	635.87	408.22	408.37	401.09
RanReal480_15	514720.84	<b>516017.98</b>	482.10	408.07	365.89	375.96
RanReal480_16	547296.05	<b>548276.15</b>	682.57	590.34	371.18	366.12
RanReal480_17	534975.71	<b>536655.06</b>	719.18	402.38	353.90	389.54
RanReal480_18	523253.63	<b>524650.86</b>	720.31	494.86	348.96	379.04
RanReal480_19	519700.55	<b>521180.84</b>	564.17	550.77	378.77	405.61
RanReal480_20	515947.56	<b>517261.92</b>	567.91	530.58	325.18	398.10
#Better	0	40				
#Equal	0	0				
#Worse	40	0				
<i>p-value</i>		2.54e-10				

547 a basin of attraction to another basin.

548 In order to assess the impact of this diversified stage on the performance of  
549 the IVNS algorithm, we created a variant of the IVNS method (denoted by  
550 IVNS-D) by removing the Shake operation of line 18 of Algorithm 1 while  
551 keeping other components of IVNS unchanged. We ran IVNS-D and IVNS  
552 20 times to solve each instance. The experimental results are summarized in  
553 Table 7, where the symbols have the same meanings as those in the previous  
554 tables.

555 Table 7 indicates that IVNS-D deteriorates the results of IVNS. First, IVNS-D  
556 performs worse than IVNS on all instances in terms of the average objective

557 value. Second, concerning the standard deviation ( $\sigma$ ) of the objective value,  
 558 IVNS obtained a better result for all instances. This experiment confirms  
 559 the usefulness of the additional diversification stage introduced in line 18 of  
 560 Algorithm 1.

561 4.4 Sensitivity Analysis of Parameters

Table 8  
 Sensitivity analysis of the parameter  $m$ . Each instance was independently solved 20 times by the IVNS algorithm for each parameter value in the range  $\{4, 6, 8, 10, 12, 14, 16, 18\}$ , and the average objective values ( $f_{avg}$ ) over 20 runs are respectively reported.

Instance/ $m$	$f_{avg}$							
	4	6	8	10	12	14	16	18
RanReal240_01	224747.89	224743.77	224750.51	224736.19	224734.17	224784.25	224725.39	224718.09
RanReal240_02	204400.52	204403.15	204425.39	204459.24	204400.94	204418.82	204448.37	204449.36
RanReal240_03	198657.99	198717.34	198685.34	198691.63	198679.41	198658.77	198661.33	198685.05
RanReal240_04	225204.66	225140.01	225146.45	225190.13	225152.00	225194.58	225242.84	225239.61
RanReal240_05	195275.00	195287.90	195237.80	195259.40	195256.56	195199.85	195246.73	195246.16
RanReal240_06	216533.35	216475.23	216454.28	216497.78	216513.23	216465.82	216515.60	216491.18
RanReal240_07	209098.93	209062.00	209060.14	209043.44	209092.56	209024.46	209021.64	209029.07
RanReal240_08	204914.37	204947.69	204883.38	204950.99	204938.25	204962.13	204943.61	204930.28
RanReal240_09	208826.26	208832.55	208883.41	208833.99	208799.52	208794.97	208822.74	208745.94
RanReal240_10	192736.25	192755.01	192815.78	192698.13	192811.66	192776.03	192640.29	192796.48
RanReal240_11	204478.83	204444.22	204464.45	204448.14	204470.10	204495.97	204478.49	204454.86
RanReal240_12	200921.94	200878.36	200830.81	200867.36	200808.50	200783.89	200887.89	200824.69
RanReal240_13	202094.54	202050.52	202042.04	202105.03	201987.09	202076.88	202094.15	202098.55
RanReal240_14	228474.87	228487.74	228483.37	228478.88	228531.74	228528.80	228525.70	228557.20
RanReal240_15	190924.82	190924.06	190907.09	190958.90	190939.27	190933.87	190944.36	190913.93
RanReal240_16	203763.28	203789.09	203798.46	203856.97	203816.01	203850.67	203746.08	203801.18
RanReal240_17	195115.48	195125.04	195150.36	195147.21	195078.97	195106.20	195132.90	195141.27
RanReal240_18	194947.69	194852.90	194893.01	194756.48	194875.62	194852.77	194866.09	194836.40
RanReal240_19	198962.52	199040.68	198966.06	198984.75	198934.98	199008.06	198988.46	198974.27
RanReal240_20	212074.56	212092.28	212056.10	212032.47	211998.49	212046.46	212001.59	212105.61
Average	205607.69	205602.48	205596.71	205599.85	205590.95	205598.16	205596.71	205601.96

Table 9  
 Sensitivity analysis of the parameter  $\beta_{max}$ . Each instance was independently solved 20 times by the IVNS algorithm for each parameter value in the range  $\{5, 10, 15, 20, 25, 30, 35, 40\}$ , and the average objective values ( $f_{avg}$ ) over 20 runs are respectively reported.

Instance/ $\beta_{max}$	$f_{avg}$							
	5	10	15	20	25	30	35	40
RanReal240_01	224676.47	224758.30	224738.12	224753.19	224749.78	224769.53	224739.29	224675.94
RanReal240_02	204384.58	204428.48	204454.16	204422.07	204403.51	204425.57	204381.89	204378.07
RanReal240_03	198621.22	198612.88	198694.68	198622.62	198700.29	198600.93	198693.71	198489.30
RanReal240_04	225111.94	225215.51	225217.08	225262.81	225203.17	225225.99	225039.70	225189.87
RanReal240_05	195202.16	195260.55	195222.28	195278.64	195256.13	195158.88	195199.11	195273.42
RanReal240_06	216421.63	216540.77	216524.29	216511.22	216500.76	216531.96	216491.43	216397.47
RanReal240_07	208954.86	209051.89	209053.73	209066.47	209030.62	209042.97	209025.39	208978.27
RanReal240_08	204957.91	204935.06	204950.87	204998.23	204918.30	204968.93	204864.60	204754.79
RanReal240_09	208775.98	208872.97	208803.31	208762.89	208814.66	208793.45	208786.65	208819.28
RanReal240_10	192729.71	192761.86	192767.85	192774.27	192787.08	192666.16	192647.02	192789.16
RanReal240_11	204457.23	204464.00	204483.40	204492.54	204490.35	204487.99	204463.07	204385.61
RanReal240_12	200897.62	200913.55	200930.03	200918.37	200868.58	200824.06	200865.56	200857.78
RanReal240_13	202110.38	202121.17	202114.62	202100.76	202097.89	202124.86	202087.71	202049.26
RanReal240_14	228454.78	228514.51	228484.05	228516.60	228536.93	228490.37	228494.82	228413.24
RanReal240_15	190869.97	190940.52	190895.56	190944.22	190966.06	190940.83	190868.54	190804.69
RanReal240_16	203702.06	203824.85	203821.77	203859.91	203778.12	203774.10	203819.94	203790.17
RanReal240_17	195096.28	195211.71	195161.74	195155.80	195166.45	195159.81	195051.22	194973.69
RanReal240_18	194861.25	194879.05	194882.91	194900.61	194900.42	194861.18	194881.60	194746.11
RanReal240_19	199005.21	199037.12	199047.45	198990.69	198983.59	199071.70	198893.53	198805.32
RanReal240_20	211997.53	212060.94	212059.02	212049.21	212060.15	212052.58	211986.43	211926.56
Average	205564.44	205620.28	205615.35	205619.06	205610.64	205598.59	205564.06	205524.90

562 Our IVNS algorithm employs two main parameters, i.e.,  $m$  and  $\beta_{max}$ . Pa-

parameter  $m$  is employed in the EVND procedure (Section 2.4.3) to control the exploitation balance between the different neighborhoods, a larger value of  $m$  leading to a more balanced neighborhood exploitation. Parameter  $\beta_{max}$  is used to control the strength of intensification search, a larger value of  $\beta_{max}$  implying a stronger intensification for the IVNS algorithm. In this section we show a sensitivity analysis of these two key parameters, which also helps to find an appropriate value for each of them.

In this study, we carried out two additional experiments based on 20 *RanReal* instances with  $n = 240$ . In the first experiment, we varied the value of  $m$  within the range  $\{4, 6, 8, 10, 12, 14, 16, 18\}$  and ran the algorithm 20 times for each value of  $m$  and each instance, while keeping other parameters with their default values as shown in Table 1. The computational results are summarized in Table 8, where the second row indicates the values of  $m$ , the first column gives the names of instances, the other columns show the average objective function values over 20 independent runs ( $f_{avg}$ ) for each value of  $m$  and each instance, and the last row shows the average results over all instances. Similarly, we varied in the second experiment the value of  $\beta_{max}$  within the range  $\{5, 10, 15, 20, 25, 30, 35, 40\}$ . The computational results are summarized in Table 9, where the second row gives the values of  $\beta_{max}$ , and the other entries have the same meanings as those in Table 8.

First, we observe from Table 8 that the performance of the IVNS algorithm is not sensitive to the setting of parameter  $m$ . Specifically, for most instances the different values of  $m$  led to very similar results in terms of  $f_{avg}$ . Indeed, the relative difference between the results yielded by the different parameter values across the 20 instances is very small ( $\leq \frac{(205607.69 - 205590.95)}{205607.69} \times 100\% = 0.0081\%$ ). Hence, the default value of  $m$  was set to 10 in this work. As for  $\beta_{max}$ , Table 9 shows that for most instances the tested values led to similar results in terms of  $f_{avg}$ , with a very small relative difference between the results yielded by the different  $\beta_{max}$  values ( $\leq \frac{(205620.28 - 205524.90)}{205620.28} \times 100\% = 0.046\%$ ). These outcomes indicate that the IVNS algorithm is not sensitive to the setting of parameter  $\beta_{max}$ . Consequently, to ensure that a lasting intensified search effect when a long computational time is allowed, the default value of  $\beta_{max}$  was set to 30 in this study.

## 5 Conclusions

The capacitated clustering problem (CCP) is a general and useful model for a number of applications. It also generalizes three well-known NP-hard problems: the maximally diverse grouping problem, the graph partitioning problem, and the handover minimization problem. In this paper, we proposed the iterated variable neighborhood search (IVNS) algorithm for solving the CCP.

602 The proposed algorithm organically combines an extended variable neighbor-  
603 hood descent (EVND) method for intensification and a shake procedure for  
604 diversification.

605 The proposed algorithm was assessed on the 133 instances commonly used  
606 in the literature, and the computational results indicated that our IVNS al-  
607 gorithm significantly outperforms the state-of-the-art CCP algorithms both  
608 in terms of solution quality and computational efficiency. In particular, the  
609 proposed algorithm improved the best known results (new lower bounds) for  
610 28 out of 83 handover minimization instances, while matching the best known  
611 results for the 55 remaining instances.

612 The investigations of several essential components of the proposed algorithm  
613 shed light on the following points. First, for the CCP, the EVND method usu-  
614 ally outperforms the standard variable neighborhood descent method in terms  
615 of the local search ability, and the 2-1 exchange neighborhood  $N_3$  reinforces  
616 the intensified search capacity of the EVND method. Second, the diversifica-  
617 tion stage is essential for the proposed algorithm to reach a suitable trade-off  
618 between the diversification and intensification of the search process.

619 Based on this work, we advance some research perspectives for further im-  
620 provements. First, within the IVNS algorithm, diversification is ensured by the  
621 shake procedure as well as the shake strength. Since different degrees of diver-  
622 sification may be needed at different search stages, it would be interesting to  
623 investigate adaptive techniques able to adjust dynamically the shake strength.  
624 Moreover, to escape deep local optima, it would also be useful to study other  
625 diversification methods like random or adaptive restarts. Second, using the  
626 presented EVND method as a local optimization procedure, it may be possi-  
627 ble to devise more efficient hybrid evolutionary algorithms for the CCP. Third,  
628 the IVNS algorithm only visits feasible solutions. Meanwhile, previous studies  
629 like [8,16] showed that tunneling through feasible and infeasible regions can  
630 improve the performance of the search process. It would be relevant to study  
631 dedicated methods able to explore infeasible regions in a controlled manner.  
632 Finally, given that the basic idea of the proposed IVNS algorithm, i.e., in-  
633 tegrating organically the EVND method with multiple neighborhoods and a  
634 diversified shake procedure, is independent of the CCP, it would be interesting  
635 to examine its applicability to other grouping or clustering problems.

## 636 **Acknowledgments**

637 We are grateful to the reviewers for their valuable comments which helped us  
638 to improve the paper. This work is partially supported by the PGMO project  
639 (2013-2015, Jacques Hadamard Mathematical Foundation, Paris, France) and

640 a post-doc grant (for X.J. Lai) from the Region of Pays de la Loire (France).

## 641 References

- 642 [1] Armas J.D., Melián-Batista B., Moreno-Pérez J.A., Brito J., 2015, GVNS  
643 for a real-world rich vehicle routing problem with time windows. *Engineering*  
644 *Applications of Artificial Intelligence* 42, 45–56.
- 645 [2] Bader D.A., Meyerhenke, H., Sanders, P., Wagner D. (eds.) 2013, Graph  
646 Partitioning and Graph Clustering. 10th DIMACS Implementation Challenge  
647 Workshop. February 13-14, 2012. Georgia Institute of Technology, Atlanta, GA.  
648 Contemporary Mathematics 588. American Mathematical Society and Center for  
649 Discrete Mathematics and Theoretical Computer Science, 2013.
- 650 [3] Benlic U., Hao J.K., 2011, A multilevel memetic approach for improving graph  
651 k-partitions. *IEEE Transactions on Evolutionary Computation* 15(5), 624–642.
- 652 [4] Benlic U., Hao J.K., 2013, Hybrid metaheuristics for the graph partitioning  
653 problem. In Hybrid Metaheuristics. Studies in Computational Intelligence 434,  
654 Chapter 6, pages 157–184.
- 655 [5] Brimberg J., Mladenović N., Urošević D., 2015, Solving the maximally diverse  
656 grouping problem by skewed general variable neighborhood search. *Information*  
657 *Sciences* 295, 650–675.
- 658 [6] Brimberg J., Janićijević S., Mladenović N., Urošević D., 2015, Solving the clique  
659 partitioning problem as a maximally diverse grouping problem. *Optimization*  
660 *Letters* doi:10.1007/s11590-015-0869-4.
- 661 [7] Chen Y., Fan Z.P., Ma J., Zeng S., 2011, A hybrid grouping genetic algorithm  
662 for reviewer group construction problem. *Expert Systems with Applications* 38(3),  
663 2401–2411.
- 664 [8] Chen Y., Hao J.K., Glover F., 2016, An evolutionary path relinking approach for  
665 the quadratic multiple knapsack problem. *Knowledge-based Systems* 92, 23–34.
- 666 [9] Deng Y.M., Bard J.F., 2011, A reactive GRASP with path relinking for  
667 capacitated clustering. *Journal of Heuristics* 17(2), 119–152.
- 668 [10] Fan Z.P., Chen Y., Ma J., Zeng S., 2010, A hybrid genetic algorithmic approach  
669 to the maximally diverse grouping problem. *Journal of the Operational Research*  
670 *Society* 62, 92–99.
- 671 [11] Feo T., Khellaf M., 1990, A class of bounded approximation algorithms for graph  
672 partitioning. *Networks* 20(2) 181–195.
- 673 [12] Ferreira C.E., Martin A, Souza C.C., Weismantel R., Wolsey L.A., 1996,  
674 Formulations and valid inequalities for the node capacitated graph partitioning  
675 problem. *Mathematical Programming* 74(3), 247–266.

- 676 [13] Ferreira C.E., Martin A, Souza C.C., Weismantel R., Wolsey L.A., 1998,  
677 The node capacitated graph partitioning problem: a computational study.  
678 *Mathematical Programming* 81(2), 229–256.
- 679 [14] Gallego M., Laguna M., Martí R., Duarte A., 2013, Tabu search with strategic  
680 oscillation for the maximally diverse grouping problem. *Journal of the Operational*  
681 *Research Society* 64, 724–734.
- 682 [15] Galinier P., Boujbel Z., Fernandes M.C., 2011, An efficient memetic algorithm  
683 for the graph partitioning problem. *Annals of Operations Research* 191(1), 1–22.
- 684 [16] Glover F., Hao J.K., 2011, The case for strategic oscillation. *Annals of*  
685 *Operations Research* 183(1), 163–173.
- 686 [17] Hansen, P., Mladenović N., Perez, J.A.M., 2010, Variable neighbourhood search:  
687 methods and applications. *Annals of Operations Research* 175, 367–407.
- 688 [18] Hendrickson B., Kolda T.G., 2000, Graph partitioning models for parallel  
689 computing. *Parallel Computing* 26(12), 1519–1534.
- 690 [19] Johnes J., 2015, Operational Research in education. *European Journal of*  
691 *Operational Research* 243(3), 683–696.
- 692 [20] Johnson E.L., Mehrotra A., Nemhauser G.L., 1993, Min-cut clustering.  
693 *Mathematical Programming* 62(1-3), 133–151.
- 694 [21] Lai X.J., Hao J.K., 2016, Iterated maxima search for the maximally diverse  
695 grouping problem. *European Journal of Operational Research* 254(3), 780–800.
- 696 [22] Lewis M., Wang H.B., Kochenberger G., 2014, Exact solutions to the capacitated  
697 clustering problem: A comparison of two models. *Annals of Data Science* 1(1), 15–  
698 23.
- 699 [23] Martínez-Gavara A., Campos V., Gallego M., Laguna M., Martí R., 2015,  
700 Tabu search and GRASP for the capacitated clustering problem. *Computational*  
701 *Optimization and Applications* 62(2), 589–607.
- 702 [24] Mladenović N., Hansen P., 1997, Variable neighborhood search. *Computers &*  
703 *Operations Research* 24(1), 1097–1100.
- 704 [25] Mladenović N., Todosijević R., Urošević D., 2016, Less is more: Basic variable  
705 neighborhood search for minimum differential dispersion problem. *Information*  
706 *Sciences* 326, 160–171.
- 707 [26] Morán-Mirabal L.F., González-Velarde J.L., Resende M.G.C., Silva R.M.A.,  
708 2013, Randomized heuristics for handover minimization in mobility networks.  
709 *Journal of Heuristics* 19(6), 845–880.
- 710 [27] Özsoy F.A., Labbé M., 2010, Size-constrained graph partitioning polytopes.  
711 *Discrete Mathematics* 310(24), 3473–3493.
- 712 [28] Palubeckis G., Ostreika A., Rubliauskas D., 2015, Maximally diverse grouping:  
713 an iterated tabu search approach. *Journal of the Operational Research Society* 66,  
714 579–592.

- 715 [29] Rodriguez F.J., Lozano M., García-Martínez C., González-Barrera J.D., 2013,  
716 An artificial bee colony algorithm for the maximally diverse grouping problem.  
717 *Information Sciences* 230(1), 183–196.
- 718 [30] Soper A.J., Walshaw C., Cross M., 2004, A combined evolutionary search  
719 and multilevel optimization approach to graph-partitioning. *Journal of Global*  
720 *Optimization* 29(2), 225-241.
- 721 [31] Urošević D., 2014, Variable neighborhood search for maximum diverse grouping  
722 problem. *Yugoslav Journal of Operations Research* 24(1), 21–33.
- 723 [32] Villegas J.G., Prins C., Prodhon C., Medaglia A.L., Velasco N., 2010,  
724 GRASP/VND and multi-start evolutionary local search for the single truck and  
725 trailer routing problem with satellite depots. *Engineering Applications of Artificial*  
726 *Intelligence* 23(5), 780–794.
- 727 [33] Weitz R., Lakshminarayan S., 1997, An empirical comparison of heuristic and  
728 graph theoretic methods for creating maximally diverse groups, VLSI design, and  
729 exam scheduling. *Omega* 25(4), 473–482.

Table 10

**Appendix:** Computational results of IVNS on the small handover minimization instances. The results are given in the form of minimization to make a direct comparison with the best known results in the literature.

Instance	BKS	IVNS				
		$f_{best}$	$f_{avg}$	$f_{worst}$	$\sigma$	$time_{avg}(s)$
20_5_270001	540	540	540.00	540	0.00	0.00
20_5_270002	54	54	54.00	54	0.00	0.00
20_5_270003	816	816	816.00	816	0.00	0.00
20_5_270004	126	126	126.00	126	0.00	0.00
20_5_270005	372	372	372.00	372	0.00	0.00
20_10_270001	2148	2148	2148.00	2148	0.00	0.00
20_10_270002	1426	1426	1426.00	1426	0.00	0.00
20_10_270003	2458	2458	2458.00	2458	0.00	0.00
20_10_270004	1570	1570	1570.00	1570	0.00	0.00
30_5_270001	772	772	772.00	772	0.00	0.00
30_5_270002	136	136	136.00	136	0.00	0.00
30_5_270003	920	920	920.00	920	0.00	0.00
30_5_270004	52	52	52.00	52	0.00	0.00
30_5_270005	410	410	410.00	410	0.00	0.00
30_10_270001	3276	3276	3276.00	3276	0.00	0.00
30_10_270002	1404	1404	1404.00	1404	0.00	0.00
30_10_270003	2214	2214	2214.00	2214	0.00	0.00
30_10_270004	2150	2150	2150.00	2150	0.00	0.00
30_10_270005	2540	2540	2540.00	2540	0.00	0.00
30_15_270001	6178	6178	6178.00	6178	0.00	0.00
30_15_270002	4042	4042	4042.00	4042	0.00	0.00
30_15_270003	4126	4126	4126.00	4126	0.00	0.00
30_15_270004	3920	3920	3920.00	3920	0.00	0.00
40_5_270001	610	610	610.00	610	0.00	0.00
40_5_270002	136	136	136.00	136	0.00	0.00
40_5_270003	234	234	234.00	234	0.00	0.00
40_5_270004	232	232	232.00	232	0.00	0.02
40_5_270005	774	774	774.00	774	0.00	0.00
40_10_270001	4544	4544	4544.00	4544	0.00	0.00
40_10_270002	2068	2068	2068.00	2068	0.00	0.00
40_10_270003	2090	2090	2090.00	2090	0.00	0.00
40_10_270004	1650	1650	1650.00	1650	0.00	0.01
40_10_270005	4316	4316	4316.00	4316	0.00	0.00
40_15_270001	8646	8646	8646.00	8646	0.00	0.01
40_15_270002	4586	4586	4586.00	4586	0.00	0.03
40_15_270003	5396	5396	5396.00	5396	0.00	0.01
40_15_270004	4800	4800	4800.00	4800	0.00	0.00
40_15_270005	6272	6272	6272.00	6272	0.00	0.00