Randomized Heuristics for the Capacitated Clustering Problem

ANNA MARTÍNEZ-GAVARA
Departamento de Estadística e Investigación Operativa,
Universidad de Valencia, Spain
Ana.Martinez-Gavara@uv.es

DARIO LANDA-SILVA
ASAP Research Group, School of Computer Science,
University of Nottingham, UK
Dario.Landasilva@nottingham.ac.uk

VICENTE CAMPOS
Departamento de Estadística e Investigación Operativa,
Universidad de Valencia, Spain
Vicente.Campos@uv.es

RAFAEL MARTÍ
Departamento de Estadística e Investigación Operativa,
Universidad de Valencia, Spain
Rafael.Marti@uv.es

May 2017.

ABSTRACT

In this paper, we investigate the adaptation of the Greedy Randomized Adaptive Search Procedure (GRASP) and Iterated Greedy methodologies to the Capacitated Clustering Problem (CCP). In particular, we focus on the effect of the balance between randomization and greediness on the performance of these multi-start heuristic search methods when solving this NP-hard problem. The former is a memory-less approach that constructs *independent* solutions, while the latter is a memory-based method that constructs *linked* solutions, obtained by partially rebuilding previous ones. Both are based on the combination of greediness and randomization in the constructive process, and coupled with a subsequent local search phase. We propose these two multi-start methods and their hybridization and compare their performance on the CCP. Additionally, we propose a heuristic based on the mathematical programming formulation of this problem, which constitutes a so-called matheuristic. We also implement a classical randomized method based on simulated annealing to complete the picture of randomized heuristics. Our extensive experimentation reveals that Iterated Greedy performs better than GRASP in this problem, and improved outcomes are obtained when both methods are hybridized and coupled with the matheuristic. In fact, the hybridization is able to outperform the best approaches previously published for the CCP. This study shows that memory-based construction is an effective mechanism within multi-start heuristic search techniques.

Keywords: Capacitated Clustering, GRASP, Matheuristic, Graph partitioning.

1. Introduction

Multi-start heuristic procedures were originally conceived as a way to exploit local or neighborhood search, by simply apply the search multiple times starting from different random initial solutions. Modern multi-start heuristic methods for combinatorial optimization problems usually incorporate a powerful form of diversification in the generation of solutions to help overcome local optimality. Without this diversification, such methods can become confined to a small region of the solution space, making it difficult, if not impossible, to find a global optimum. Most of such methods perform these steps iteratively: apply a randomized constructive method followed by a local search procedure. In these methods, diversification comes from the iterative randomized construction of solutions.

Multi-start heuristic methods for combinatorial optimization can be classified as suggested by Martí et al. (2013) in memory-based and memory-less procedures. GRASP (Greedy Randomized adaptive Search procedure) is probably the best well-known memory-less multi-start heuristic method (Resende and Ribeiro 2010), while tabu search (Glover and Laguna 1997) is nowadays a reference for memory based approaches. In this paper, we focus on both memory-based and memory-less multi-start heuristic methods, and investigate the effect of randomization in these designs. We use the Capacitated Clustering Problem (CCP), an NP-hard combinatorial optimization problem, as a test case for our proposals and findings. This paper is the second part of our study on the CCP, initiated in Martínez-Gavara et al. (2015), with two main objectives. The first one is to improve the results of the previous methods. The second objective is to compare memory-based with randomized constructive algorithms.

Many constructive heuristic algorithms for combinatorial optimization build a solution incrementally, by adding at each step, an element to the partial solution under construction. The way in which this element is selected, constitutes the distinguishing component of the constructive method. In this paper, we are interested in the interaction between greediness and randomization within the constructive method. Since there is no guarantee that a greedy randomized approach will produce a solution that is locally optimal, local search if often applied after the construction step in an attempt to find an improved solution that is also locally optimal. This was first proposed by Feo and Resende (1989) for the Set Covering Problem and was later referred to as GRASP.

Strategic oscillation (SO) is a methodology closely linked to the origins of tabu search, and operates by directing the local search moves in relation to a critical level identified in the construction stage. This methodology provides an interesting alternative to improve traditional constructive approaches. In particular, we consider a constructive/destructive type of strategic oscillation, where constructive steps "add" elements and destructive steps "drop" elements from the solution. The alternation of constructive and destructive steps is a successful strategy to enhance of such traditional constructive procedures. In this paper, we focus on a simplified and effective SO method known as Iterated Greedy (IG). This method generates a sequence of solutions by iterating over a greedy constructive heuristic using two main phases: destruction and construction. IG is a memory-based multi-start easy to implement that has exhibited state-of-the-art performance in some settings (Ruiz and Stützle, 2008).

In this paper, we investigate these two successful methodologies in multi-start methods: GRASP and Iterated Greedy, and their hybridization. The former constructs *independent* solutions, while the latter can be viewed as a constructive method of *linked* solutions. These are two very different approaches to construct a solution. Both methods combine greediness and randomization in different ways. The aim of this investigation is to identify ways to exploit better greediness and randomization. For our experiments, we consider the CCP, which is a difficult optimization problem. However, our objective is to identify effective strategies and patterns that could succeed in other settings. Hence, the intended contribution of this paper is to exploit greediness and randomization within the context of multi-start heuristic search effectively. In a broader sense, we can say that we are comparing memory-less and memory-based designs within constructive methods.

We complete this introduction with the next subsection where the CCP is described both in general and in the context of the handover minimization in mobility networks, and illustrate it with an example. Section 2 describes the solution methods for this problem. It is divided into several subsections, where the first one, Subsection 2.1, is devoted to previous GRASP methods for the CCP. The main contributions of our paper are described in the following subsections. In particular, Subsection 2.2 describes a new 2-1 neighborhood to improve solutions, while Subsection 2.3 describes the destructive algorithms for IG. GRASP designs only incorporate constructive neighborhoods but, as mentioned, IG incorporates the notion of destructing a solution. Finally, Subsection 2.4 is devoted to our IG algorithms, including a hybrid heuristic of both previous methods called IG-GRASP for which we also adapt the filtering mechanism introduced in Laguna and Martí (1999) to make the search more efficient. Our final contribution is a post-processing based on the integer linear programming formulation of the CCP. Section 3 describes how we enhance the standard formulation by adding valid inequalities adapted from the literature. In particular, we propose a matheuristic procedure that in some cases is able to improve the best solution found with the hybridized heuristic by solving the enhanced formulation on a fraction of the original variables. The paper finishes with a computational

study with extensive experimentation, which reveals the contribution of the memory-based elements, and the associated conclusions.

1.2 The Capacitated Clustering Problem

The problem of maximizing diversity deals with selecting a subset of elements from a given set in order to maximize the diversity among the selected elements; see Glover et al. (1995). Several models have been proposed to deal with this combinatorial optimization problem. All of them require a diversity measure, typically based on a distance function. The definition of this distance between elements is customized to specific applications. As described in Glover et al. (1998), maximization diversity models have applications in plant breeding, social problems, ecological preservation, pollution control, product design, capital investment, workforce management, curriculum design, and genetic engineering. The most studied model related to diversity is probably the Maximum Diversity Problem (MDP) also known as the Max-Sum Diversity Model (Ghosh 1996), in which the objective is to maximize the sum of the distances between the selected elements. The Max-Min Diversity Problem (MMDP), in which the minimum distance between the selected elements is maximized, has been also well documented in recent studies; see Resende et al. (2010).

In this paper we consider an interesting variant within the diversity models. The aim of the Capacitated Clustering Problem (CCP) is to find a partition of the set of points into different groups in order to maximize some weighted measure of the distance among the points in the same group. Early developments on clustering were devoted to a different variant. Osman and Christofides (1994) introduced a variant of the clustering problem where the objective is to minimize the total scatter of objects from the 'centre' of the cluster to which they have been allocated. A simple constructive heuristic, a λ-interchange generation mechanism, a hybrid simulated annealing (SA) and tabu search (TS) algorithm which has computationally desirable features using a new non-monotonic cooling schedule, were proposed. França et al. (1999) followed upon the same variant in which the objective is to find p customers, called medians, from which the sum of the distances to all other customers in the cluster is minimized. In this article, an adaptive tabu search approach is applied to solve the problem. More recently, Scheuerer and Wendolsky (2006) developed a scatter search algorithm for the same problem. Chaves and Lorena (2010) considered a different variant in which each cluster has a center but they maximize the diversity with respect to this center. Other successful hybridizations of the tabu search and simulated annealing methodologies are Swarnkar and Tiwari (2004) and Mishra et al. (2008).

One of the most recent applications of the CCP can be found in the context of facility planners at mail processing and distribution centers within the US Postal Service. In particular, in the design of the zones to help rationalize the bulk movement of mail, see Deng and Bard (2011). Morán-Mirabal et al. (2013) tackled a very interesting real-world problem, which as shown in Martínez-Gavara et al. (2015) turns out to be an application of the CCP. Hence, we use this real-world application in the context of mobility networks for the investigation presented here.

Any mobile transceiver such as cell phones, tablets, portable computers, etc. needs a radio signal for communication among such devices. Furthermore, these mobile transceivers move between areas or cells that are covered by fixed base stations, and they may need to connect over time to several different base stations. The transfer of connection from one base station to another is called a

handover. A mobility network (see Fig. 1) also contains some radio network controllers (RNC), which control many of the base station operations, including traffic and handover. Handovers between base stations connected to different RNCs tend to fail more often than handovers between base stations connected to the same RNC. Handover failures result in dropped connections and therefore should be minimized. To sum up, the Handover Minimization Problem consists to assign towers to RNCs such that RNC capacity is not violated and the number of handovers between base stations connected to different RNCs is minimized. The set of base stations assigned to a RNC can be viewed as a cluster, and the minimization of handovers between different clusters is equivalent to the maximization of handovers within the same cluster. Therefore, this problem is equivalent to the CCP.

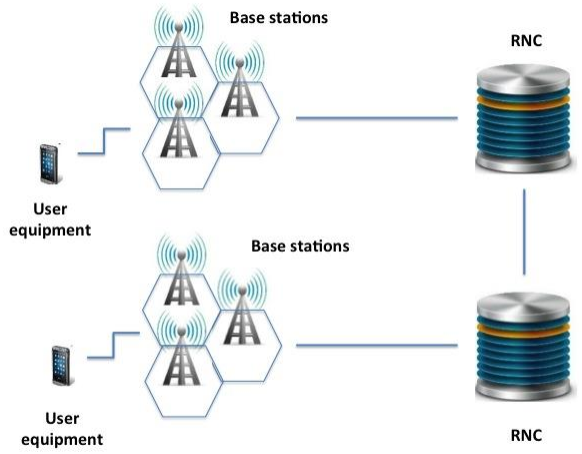

Fig 1. Typical Mobility Network.

We wish to partition a set V of n nodes into p clusters such that the sum of benefits c_e, of edges $e \in E$ within each cluster is maximized, and the sum of the node weights, $w_i \geq 0$ of nodes $i \in V$ within the same cluster is within some integer capacity limits, L and U.

Figure 2 shows a small example of a CCP with nine nodes and benefits associated with the edges. We consider three clusters with a capacity bound between 3 and 5 for each one. Nodes are numbered from 0 to 8. Assume that the node weights are $w_i = 1$ for $i \in \{0,1,4,5,6,7,8\}$, $w_2 = 2$ and $w_3 = 3$.

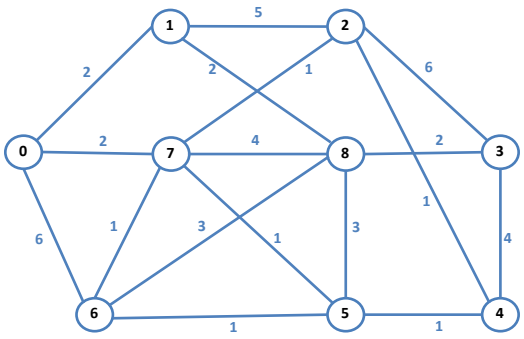

Fig 2. Small example.

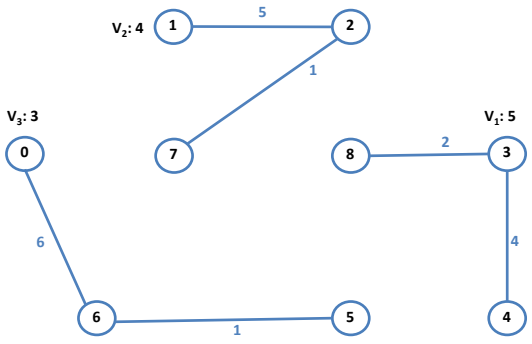

Fig 3. Example of feasible solution to a CCP.

Figure 3 shows a feasible solution to this CCP. It consists of three clusters: $V_1 = \{3,4,8\}, V_2 = \{1,2,7\}$ and $V_3 = \{0,5,6\}$, where the sum of the node weights within each cluster is 5, 4 and 3, respectively. This feasible solution has an overall benefit of 19.

2. Methods

In this section, we first describe the three previous GRASP methods proposed for the CCP, and then describe our new methods. Note that these three GRASP methods implemented a local search based on exchange and insertion moves. We propose a new GRASP in which the local search implements a 2-1 move (two elements are exchanged with a single element from another cluster).

2.1 Previous GRASP methods

Three different GRASP methods have been proposed for the Capacitated Clustering Problem:

- PrevGRASP1: Deng et al. (2011) proposed a GRASP with a post-processing stage by using Path-Relinking.
- PrevGRASP2: Morán-Mirabal et al. (2013) also applied GRASP with path-relinking and included the variant known as evolutionary path-relinking.
- PrevGRASP3: Martínez-Gavara et al. (2015) proposed a simplified GRASP that provides high-quality solutions in short computing times.

In the construction phase of the PrevGRASP1 (Deng and Bard, 2011), the p clusters are first seeded with the heaviest weight edges algorithm (HWE), and then completed with a greedy randomized procedure. Specifically, the HWE identifies the p nodes with the largest weights and assigns them, separately, to the p clusters. The heaviest edges incident to these nodes are then identified, and their endpoints are assigned to the corresponding clusters. An alternative constructive method, labeled CMC, also proposed by Deng and Bard (2011) was shown to be inferior to HWE and therefore we do not consider it here.

Let us consider the example in Figure 2 to illustrate the behavior of the HWE algorithm. HWE produces clusters containing two nodes. Initially, the heaviest node (node 3) is assigned to cluster V_1, while the second heaviest node (node 2) is assigned to cluster V_2. As long as the remaining nodes have all the same weight, cluster 3 is left empty. In the next step, the heaviest unassigned edge incident with each cluster is assigned to it. Thus, edge (3,4), with a weight of 4, is assigned to cluster V_1, similarly edge (1,2) is placed into V_2. Finally, the heaviest unassigned edge (1,7) is then assigned to V_3. At this point, a candidate list CL of elements is built to continue the construction process according to the GRASP methodology. In particular, CL is formed with the nodes and edges (pairs of nodes) that can be inserted into a solution cluster without exceeding the upper capacity limit U. At the end of the construction phase the three clusters are $V_1 = \{3,4,8\}, V_2 = \{1,2,7\}$ and $V_3 = \{0,5,6\}$.

In the second phase of PrevGRASP1, the authors used three different neighborhoods to improve a constructed solution x: $N_1(x)$, $N_2(x)$, and $N_3(x)$. Let V_k be the set of nodes in cluster k of this solution, and let W_k be the sum of the weights of the nodes in V_k (i.e., $W_k = \sum_{i \in V_k} w_i$); then, W_k must be within the capacity limits: $L \leq W_k \leq U$ for $k = 1, 2, \ldots, p$. $N_1(x)$ is the result of extended insertion moves, which consider temporarily infeasible moves. If node 7, in the example shown in Figure 3, is

moved from cluster V_2 to cluster V_1 but there is not enough capacity in cluster V_1 for the node 7 (weight of node 7 is 1, and the sum of the weights in cluster V_1 is 5). Then, instead of discarding the move, node 4 in V_1 could be moved from cluster 1 to cluster V_3, which has enough capacity for including it, and then node 7 is placed in cluster V_1. In addition, the extended insertion move is feasible. Applying it to the solution from the construction phase, this movement $N_1(x)$ produces the solution of Figure 4.

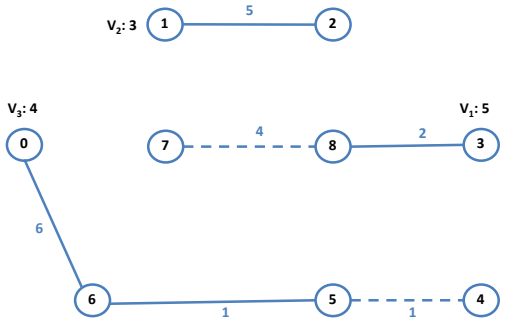

Fig 4. Neighborhood $N_1(x)$ in PrevGRASP1.

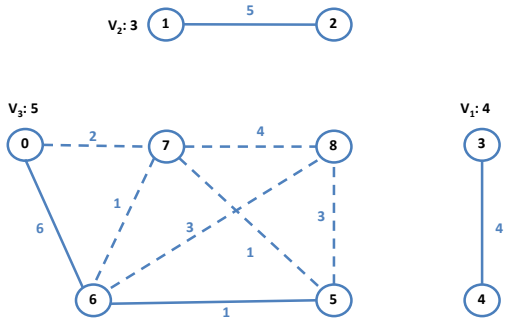

Fig 5. Neighborhood $N_2(x)$ in PrevGRASP1.

$N_2(x)$ consists of edge insertions. Given an edge $(i, j) \in E$, two cases may arise; both nodes i and j are in the same cluster, or the edge spans two clusters. The edge insertion considers moving both nodes to another cluster, as long as the resulting solution remains feasible. In the example case shown in Figure 3, edge (7, 8) spans two different clusters, V_1 and V_2. The resulting solution considers moving both nodes to cluster V_3, obtaining the graph shown in Figure 5 in which dashed edges carry additional benefits. Only capacity-feasible moves are considered. Finally, $N_3(x)$ implements a classical swap move, that is, one in which a node i is moved from a cluster k to a cluster s, and simultaneously a node j is moved from the cluster s to the cluster k. As in the other neighborhoods, the move is performed only if the resulting solution is feasible.

In their computational experiments, Deng and Bard (2011) compared their designs and concluded that the combination of HWE with Randomized Variable Neighborhood Descent (RVND) resulted in the best overall performance. In this improvement method the neighborhood to be searched in the next iteration is probabilistically selected, where the probability of selection is linked to the merit of each neighborhood as determined by the quality of the solutions found during the search. We use this variant for the purpose of comparison later in this paper.

In PrevGRASP2 (Morán-Mirabal et al. 2013) a GRASP with Path Relinking is proposed for the handover minimization in mobile networks problem, which as discussed above, is equivalent to the CCP. A randomized greedy algorithm constructs a solution one by assigning one base station (node) to an RNC (cluster) one at a time. RNCs are initially permuted at random and the algorithm scans the RNCs in the permutation order, dealing with only one RNC at a time. Let k be the current RNC being scanned. Base stations are assigned to the RNCs while they have available capacity. After scanning all available RNCs, it may occur that not all base stations are assigned. In such a case, a repair procedure is applied to seek feasibility.

Once the randomized greedy construction method produces an assignment vector, a local search algorithm attempts to improve the assignment by making changes on it. Specifically, Morán-Mirabal et al. (2013) proposed three local search algorithms, move-1, move-max, and swap-2. The three algorithms scan the base stations in increasing order of their total traffic (capacity of the nodes). For base station i, the procedure move-1 checks if there is any other RNC with enough capacity to accommodate i such that the reassignment from its current RNC to the other one reduces the total handover count. If such RNC is found, base station i is reassigned to it. In terms of the CCP, maximizes the sum of the benefits within each cluster. Consider the initial feasible solution $V_1 = \{3,4,8\}, V_2 = \{1,2,7\}$ and $V_3 = \{0,5,6\}$, shown in Figure 3, then the move-1 reassigns node 8 from V_1 to V_3, obtaining a total benefit of 23 instead of 19, which is equivalent to reduce the total handover count from 26 to 22, see Figure 6. In the case of move-1, the procedure is restarted at the first base station in the permutation (i.e. the base station with the smallest traffic), whereas in the case of move-max it proceeds to the next station in the permutation (i.e. the station with least traffic among those with more traffic than the just reassigned station). After scanning all base stations without finding any improving move, the procedure ends. In the case of swap-2, pairs of base station assignments are considered for swapping.

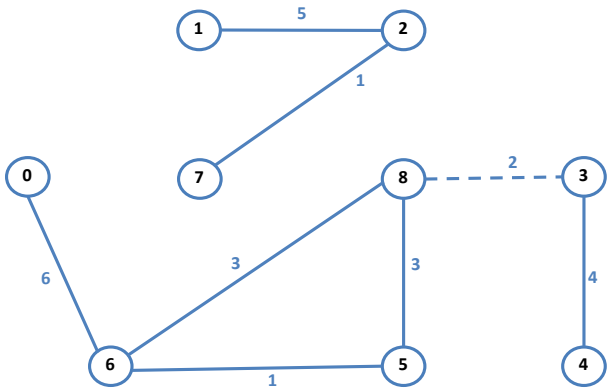

Fig 6. Move-1 in PrevGRASP2.

Finally, in PrevGRASP3 (Martínez-Gavara et al. 2015) a GRASP implementation is proposed in which only nodes are candidates in the construction process and two simple neighborhoods are combined into a deterministic Variable Neighborhood Descent (VND) design. The GRASP method starts by seeding the p clusters $V_1, V_2, \ldots, V_p$ with p randomly selected nodes. Then, the clusters are explored in lexicographical order assigning elements until all of them satisfy the lower bound constraint. In the example shown in Figure 1, the clusters are initialized by the seeds $V_1 = \{4\}, V_2 = \{2\}, V_3 = \{0\}$. For the first cluster, the candidate list is formed with all the unassigned nodes and the value $I(i,1)$ is calculated for all pairs $(i,1)$ of nodes and clusters 1. RCL_1—that is, the restricted candidate list of nodes for cluster 1—is formed with all nodes i for which $I(i,1)$ is within a percentage $\alpha \in]0,1]$ of the maximum value $I_{max} = \max_{i \in CL} I(i,k)$ in CL. Then the method selects randomly an element in RCL_1, and performs the corresponding assignment. In this case simply example $RCL_1 = \{3\}$, then node 3 is added to cluster 1. It is proceed in a similar way for the all clusters. Figure 7 shows the partial solution obtained at the end of this phase.

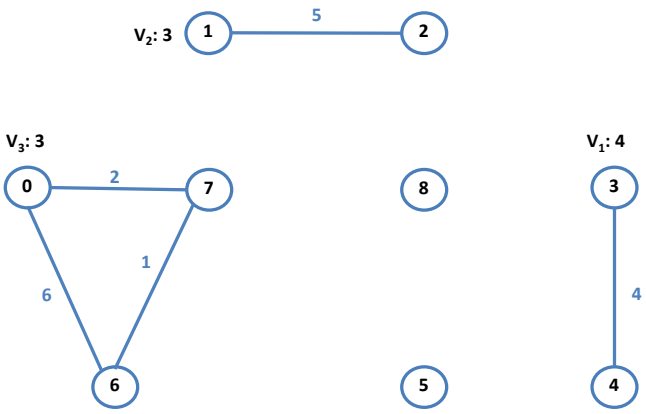

Fig 7. Initial partial solution in PrevGRASP3.

The next step consists of assigning all the unassigned nodes to those clusters such that the sum of the weights of the elements plus the weight of the new node is lower than or equal to the upper bound U, in our example, 5. The candidate list CL is formed with the pairs (i,k) with unassigned nodes i and those clusters k such that the solution remains feasible. The method proceeds to evaluate $I(i,k)$ for all (i,k) in CL, build RCL with the (i,k) pairs with an evaluation within a percentage $\alpha \in]0,1]$ of the maximum value in CL, and select one pair at random. It stops when all the nodes have been assigned to clusters. In the example above, suppose that the clusters are set by $V_1 = \{3,4\}, V_2 = \{1,2\}, V_3 = \{0,6,7,8\}$, so only the node 5 is left. In that case, the CL is formed by node 5 to cluster V_3 with a contribution of 5, to cluster V_1 with a contribution of 1 and no contribution to cluster V_2. This constructive method is denoted by CM.

Once a solution x is obtained, the improvement method consists of a deterministic VND based on two neighborhoods, $N_0(x)$ and $N_3(x)$. The method determines first a best neighbor x' of x in $N_0(x)$. If x' is better than x, then x is replaced with x' and the method searches now for the best neighbor in $N_0(x')$, thus performing a local search in N_0 while it improves the current solution. When the current solution x cannot be improved in N_0, then the method resorts to N_3 and determines the best neighbor x' of x in $N_3(x)$. If x' is better than x, then the method comes back to search in $N_0(x')$; otherwise the VND finishes. In short, the algorithm performs a local search for the best solution in N_0 and only resorts to searching N_3 when the process is trapped in a local optimum found in N_0. The improvement method considers only feasible moves.

2.2 A new 2-1 Neighborhood

The neighborhood $N_4(x)$ explores the exchange of two nodes, say i and j, in the same cluster k with a node l in another cluster s. This move (Martínez-Gavara et al. 2015) can be simply called a 2-1 exchange, and it makes possible to swap nodes that individually are not allowed for reasons of capacity, as illustrated below. We propose a GRASP, called GRASP2-1, in which the constructive method is the one in PrevGRASP3 but the improvement method performs 2-1 exchanges.

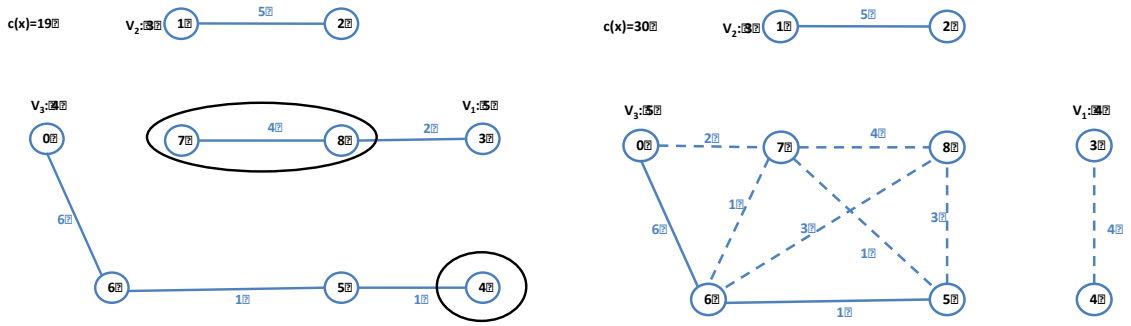

Fig 8. Example of a 2-1 exchange in neighborhood $N_4(x)$.

Consider the example illustrated in Figure 8-left. Cluster V_1 contains nodes 7 and 8 with weights $w_7 = w_8 = 1$ and does not have any remaining capacity. Cluster V_3 contains node 4 with $w_4 = 1$ and it has a remaining capacity of 1. The 2-1 exchange moves nodes 7 and 8 from V_1 to V_3, and node 5 from cluster V_3 to V_1, in such a way that results in a feasible solution (shown on the right part of Figure 8) with an improved objective function. In GRASP2-1, the entire neighborhood of 2-1 exchanges $N_4(x)$ is explored. For a given solution, all the 2-1 exchanges are evaluated and the best one, according to the objective function, is selected. In other words, we implement a best-of-all strategy, in which the entire neighborhood is examined and the best move is selected in each iteration of the local search.

```
1.  Let x be the solution obtained with the constructive phase
while(improve)
      for(each cluster A)
            for(each cluster B)
2.              i,j two nodes in A
3.              k one node in B
4.              m21 ← exchange i,j from A to B and k from B to A
5.              eval(i,j)k ← profit of move m21
                if eval(i,j)k provides the best value then
6.                    eval_best ← eval(i,j)k
                end
            end
      end
7.    xi ← be the solution of the eval_best .
      if xi provides better objective than x then
8.       x ← xi
      end
end
```

Algorithm 1. Local search based on $N_4(x)$, IM2-1.

It is worth mentioning that we explored the implementation of a VND post-processing method in GRASP2-1, as it is implemented in PrevGRASP3. However, preliminary results not reported here, showed the superiority of GRASP2-1 without this VND post-processing. We therefore limit GRASP2-1 to apply the improvement method described above, called IM21, to the solutions constructed with the method in PrevGRASP3 described in the previous subsection.

2.3 Destructive Neighborhoods

Our first destructive method DM1 applies a simple mechanism removing some nodes randomly from each cluster. The percentage of elements removed in each cluster is defined by the search parameter β_1. Our second destructive method, DM2, is based on a greedy mechanism. Given a feasible solution x, where V_k is the set of nodes assigned to a cluster k, for each element i in cluster k we define $I(i, V_k)$ as the contribution of node i to the objective function value in cluster k. In mathematical terms:

$$I(i, V_k) = \sum_{j \in V_k} c_{(i,j)}$$

where $c_{(i,j)}$ is the cost or benefit of the arc (i, j). Let $I(i) = \sum_{j \in V} c_{(i,j)}$ be the potential contribution of node i to the objective function. Then, the relative contribution of node i to cluster k can be computed as:

$$I_R(i, V_k) = \frac{I(i, V_k)}{I(i)}. \quad (1)$$

If this value, which is in $[0,1]$ by design, is small, it indicates that node i might increase the objective function if moved to a different cluster. Then, the candidate list of nodes in V_k to be considered for a move (CL) in the current solution is formed by

$$CL(V_k) = \{i \in V_k : I_R(i, V_k) \leq \gamma_k\}, \quad (2)$$

where

$$\gamma_k = \delta \min_{i \in V_k} I_R(i, V_k) + (1 - \delta) \max_{i \in V_k} I_R(i, V_k) \quad \text{with} \quad \delta \in [0,1]. \quad (3)$$

As shown above, the threshold γ_k is computed by means of the parameter δ, which manages how restrictive is CL. If δ is close to 1 then γ is close to the minimum of the relative contributions, and the candidate list contains a small fraction of the nodes in V_k. On the contrary, if it is close to 0, then CL contains most of the nodes in the cluster. The destructive method DM2 removes from the solution, a percentage β_2 of elements from the candidate list of each cluster.

2.4 Iterated Greedy

The Iterated Greedy method (IG) alternates between destructive and constructive phases. During the destructive phase, some elements are removed from the solution. Next, it applies a greedy constructive method to reconstruct the partial solution and obtain a new solution. Then, an acceptance criterion is applied to decide whether the new solution replaces the current solution or not. The method iterates following this pattern until a stopping criterion is met. We refer the reader to Ying et al. (2010) and Lozano et al. (2014) for descriptions of successful applications of IG. In this subsection we investigate two different IG algorithms, IG and IG-GRASP.

Our first implementation of the Iterated Greedy methodology, called simply IG1, starts from an initial solution x, built with the CM algorithm and improved with IM2-1 (see Section 2.2). Then, IG1 iteratively alternates between destructive and constructive phases. In the destructive phase, a percentage β_1 of the nodes are removed using the procedure DM1. Then, the constructive phase

applies the greedy heuristic CM to reconstruct the solution. Additionally, the local search phase IM2-1 is applied to improve the new solution. This method is shown in Algorithm 2, in which we can see the update mechanism of the incumbent solution each time a newly reconstructed solution has been obtained.

```
1.  Let x be the initial solution
2.  Let T be the maximum time allowed
3.  x_b ← x
    while(Time limit T is not reached)
4.        y ← DM1(x)
5.        y_c ← CM(y)
6.        x_i ← IM2-1(y_c)
          if x_i is better than x_b then
7.              x_b ← x_i
          end
8.        x ← x_b
    end
```

Algorithm 2. Iterated Greedy IG1.

Algorithm 3 below describes our hybridization between IG and GRASP called IG-GRASP. Initially, as in IG1, it builds a complete solution with CM and then improves it with IM2-1. Then, the algorithm iteratively applies the destructive algorithm DM2, then the constructive method CM, and finally the improvement procedure IM2-1. However, after a number of pre-established iterations ($\gamma\, n$) applying these three methods consecutively with no improvement, instead of ending the procedure (as it is the case of IG), the hybrid algorithm resorts to GRASP2-1 to generate a new solution (built from scratch) to start again.

```
1.    Let x be the initial solution
2.    Let T be the maximum time allowed
3.    Let γ n be the maximum of iteration without improving allowed
4.    x_b best solution generated
while(T is not reached)
5.    y_c ← CM(x_b)
6.    x_i ← IM2-1(y_c)
      while(l < γ n)
8.        y ← DM2(x_i)
9.        y_c ← CM(y)
10.       x_i ← IM2-1(y_c)
          if x_i provides better objective than x_b then
11.           x_b ← x_i
12.           l ← 0
          else
13.           l ← l + 1
      end
end
```

Algorithm 3. Hybridization of Iterated Greedy with GRASP (IG-GRASP).

An interesting distinction between different IG methods is in the acceptance criterion to select the solution for applying the destructive method. As described in Lozano et al. (2014), in the 'Replace if better' acceptance criterion, the new solution is accepted only if it provides a better objective function

value. In other words, the IG iterates over the best solution found. However, this can lead to stagnation situations of the search due to insufficient diversification. On the other hand, the "Always replace" acceptance criterion applies the destruction phase to the most recently visited solution, independently to its objective function value. This criterion clearly favors diversification over intensification, because it promotes a stochastic search in the space of local optima. We applied the latter one to our IG variants.

After a number of iterations, it is possible to estimate the fractional improvement achieved by the application of the improvement phase and use this information to increase the efficiency of the search (Laguna and Martí, 1999). In particular, based on the average improvement achieved by the local search in previous iterations, the filtering method discards the constructed solutions when it is unlikely that they improve the best found so far, saving the associated computation time. It is based on a search parameter λ representing a threshold on the number of standard deviations away from the estimated average percentage improvement. Preliminary experiments to test the effect of different λ values have been performed and are reported in Section 4.

3. Theory: A Matheuristic post-processing

This section first describes the standard mathematical programming formulation for the CCP. Then, we propose adaptations of valid inequalities to strength the formulation, and finally a method to use the information from the best solution found with the heuristics to fix some variables in this formulation, which permits to apply it to large size problems (as a heuristic method itself).

Let the binary variable $y_{ek} = 1$ if and only if edge e has both of its end points in cluster k, and let the binary variable $x_{ik} = 1$ if and only if node i is assigned to cluster k. The CCP can be formulated as proposed by Morán-Mirabal et al. (2013) as a mixed integer program.

$$\text{(CCP) Maximize} \quad \sum_{k=1}^{p} \sum_{e \in E} c_e y_{ek}$$

$$\text{subject to} \quad \sum_{k=1}^{p} x_{ik} = 1 \qquad \forall i \in V$$

$$y_{ek} \leq x_{ik}, \qquad \forall e = (i,j) \in E, k = 1, \dots, p$$

$$y_{ek} \leq x_{jk}, \qquad \forall e = (i,j) \in E, k = 1, \dots, p$$

$$L \leq \sum_{i=1}^{n} w_i x_{ik} \leq U \qquad \forall k = 1,2, \dots, p$$

$$x_{ik} \in \{0,1\} \qquad \forall i \in V, k = 1, \dots, p$$

$$0 \leq y_{ek} \leq 1 \qquad \forall e = (i,j) \in E, k = 1, \dots, p$$

Ferreira et al. (1998) proposed several valid inequalities for a family of clustering problems. We adapted some of them to the CCP. The first one is the so-called triangle inequality. If we denote with

$y_{ek} = y_{ijk}$ as the binary variable that takes the value 1 if edge $e = (i, j)$ is in cluster k, it is easy to see that:

$$y_{ijk} + y_{jsk} \leq 1 + y_{sik}$$

For every set of three edges $(i, j), (j, s), (s, k) \in E$. This inequality can be generalized to

$$y_{ijk} + y_{jtk} + y_{tsk} \leq 2 + y_{sik}$$

We consider the integer linear formulation above with these two families of inequalities. It is clear from previous papers that we cannot directly solve medium and large instances with this formulation to optimality. We therefore propose a heuristic to use this extended model.

The method, called Math, starts by solving the problem with a heuristic, say for example GRASP-IG, and then use the solution obtained to fix some variables in the integer linear program. This is indeed a standard method in mathematical programming: to "refine" a heuristic solution. In particular, we fix a proportion φ of the number of edges $|E|$ in the formulation according to the heuristic solution. If for example vertices 1 and 2 are in the same cluster k in the heuristic solution, we can set $x_{1k} = x_{2k} = y_{12k} = 1$, and $x_{1l} = x_{2l} = y_{12l} = 0$ for the clusters $l \neq k$. We consider the edges in E ordered according to their benefits, where the edge with the largest benefit comes first, and set the variables associated to the first $\varphi|E|$ edges to 1. It is clear that if φ is close to 1, most of the variables are fix in the model, and then it is very likely that we obtain the same heuristic solution when solving the model. On the other hand, if the proportion φ is very small, and close to 0, only a few variables are set in the model, and therefore it is unlikely to solve the model in moderate computing times. We try several values of this parameter in our computational experiments: φ= 0.1, 0.25, 0.5, 0.75, and 0.9.

4. Computational Experiments

This section describes the computational experiments that we performed to test the effectiveness and efficiency of the procedures described above: PrevGRASP1, PrevGRASP2, PrevGRASP3, and our new GRASP2-1. Additionally, we test the proposed Iterated Greedy algorithm, IG1, and the hybrid method IG-GRASP. The six methods have been implemented in C and to generate random numbers we use the `rand()` function. All experiments were performed on a 2.8 GHz Intel Core i7 with 8 GB of RAM.

We have two main objectives in this section. We first perform a preliminary testing with the objective of finding effective configurations for our methods (i.e., to fine tune their algorithmic parameters). This experimentation has important implications since the performance of the methods strongly depends on the values of the key search parameters. To prevent the over training of the algorithm, we perform these experiments on a small set of representative instances. We are looking for a robust configuration of our methods that performs well across all types of instances. Since run time is a critical factor in the heuristic domain, the goal is to find parameter values that result in an effective tradeoff between solution quality and computational effort. Once the methods are configured, we perform the final step of our experimentation: the competitive testing. The objective in this second stage is to show that our algorithm is able to obtain better solutions than the existing methods. Note that we have to adopt a statistical perspective to compare the algorithms. We consider a set of instances and run the different methods under the same conditions (same computer and total running time), and we report

average results. To generalize the results from the sample set considered to the entire set of instances of this problem (population), we apply statistical tests, which permit to draw sound conclusions.

We employed 60 CCP problem instances in our experimentation. This benchmark set of instances, referred to as CCPLIB, is available at http://www.optsicom.es/ccp. This benchmark, formed with three sets (RanReal, DB, and MM) was used and described in Martínez-Gavara et al. (2015), so we do not reproduce here its characteristics. We have selected 15 representative instances with different characteristics to perform a preliminary experimentation in order to identify effective values for parameters in our new three methods: IG, GRASP21, and IG-GRASP. Specifically, we selected 6 *RanReal* instances with $n = 240,\ p = 12,\ L = 75$, and $U = 125$, 3 *DB* instances with $n = 82,\ p = 8,\ L = 25$, and $U = 75$ and 6 *MM* instances, one of each combination (b, r) with $n = 100, 200$.

We use the following metrics to measure the merit of each procedure:

Dev Local average percent deviation from the best value.

$Best$ Fraction of instances for which a procedure is able to match the best solution.

$Score$ Fraction of the instances for which the competing procedures "win" (i.e., they produce better solutions than the other procedures being scored). This is calculated as $(q(p-1)-r)/(q(p-1))$, where p is the number of procedures being compared, q is the number of instances, and r is the number of instances in which the $p-1$ competing procedures find a better result. Hence, the best score is 1 (when $r = 0$) and the worst is 0 ($r = q(p-1)$).

In our first preliminary experiment, we test the parameter β_1 in the IG1 method, which gives the number of removed elements $\max(1, \beta_1 |V_k|)$ in each cluster V_k. This parameter is tested in the set $\{0.1, 0.3, 0.5, 0.7\}$, where the 15 representative instances are run for 60 seconds, obtaining the best value with $\beta_1 = 0.1$, as it is shown in Table 1.

$\boldsymbol{\beta_1}$	$\boldsymbol{Dev}$	$\boldsymbol{Best}$	$\boldsymbol{Score}$	$\boldsymbol{Time}$
0.1	0.31%	67%	0.67	60.03
0.3	0.59%	20%	0.67	60.07
0.5	0.70%	20%	0.60	60.09
0.7	0.83%	27%	0.27	60.16

Table 1. Best parameter values identified for the IG1 method.

In the second preliminary experiment, we test the parameter α applied in the construction algorithm of GRASP2-1 to balance randomization and greediness. We do not report the results of this experiment since we did not observe significant differences among the α-values tested. Following the selection of the other previous GRASP methods (Martínez-Gavara et al. 2015) we set the value of α to 0.6.

In our third preliminary experiment we test the three parameters of the IG-GRASP algorithm:

- $\boldsymbol{\beta_2}$, which is percentage of the removed elements in the candidate list of each cluster,
- $\boldsymbol{\delta}$, which controls the candidate list size in DM2, and
- $\boldsymbol{\gamma}$, which determines the termination criterion for the entire method.

We utilized a sequential design in which we set the value of each parameter one at a time, while the others are kept constant. Each run consisted of 60 seconds of CPU time. For the sake of simplicity, we only report here the results of the experiment to set γ (see Table 2). This table shows, as expected, a trend in which as the number of iterations increases, the quality of the solution also increases. Note however, that values larger than 0.5 do not follow this trend and obtain lower quality results. We therefore set this parameter to $\gamma = 0.5$. In a similar way, we set $\beta_2 = 0.1, \delta = 0.7$ after the experimentation.

γ	Dev	$Best$	$Score$	$Time$
0.1	0.22%	13%	0.18	60.61
0.3	0.06%	47%	0.73	62.81
0.5	0.00%	87%	0.93	64.61
0.7	0.02%	60%	0.82	64.53

Table 2. Best γ parameter values identified for the IG-GRASP method.

In our final preliminary experiment, we test the efficiency of the filtering mechanism in GRASP (Laguna and Martí, 1999). Table 3 reports the results obtained with the method when running with different values of the two filter parameters. The first one is $tmax$, the number of initial iterations for which we compute the mean and standard deviation of the improvement achieved with the local search. The second one is λ , the value to compute the filtering threshold. Specifically, this table reports Dev and Time, as in the previous experiments, and the average number of solutions discarded for improvement, # skipped, in 100 constructions.

$tmax$	λ	Dev	$\#\ skipped$	$Time$
10	-1	0.51%	85	9.80
	0	0.24%	69	20.47
	1	0.13%	41	37.74
	2	0.06%	18	55.45
	3	0.03%	6	63.71
20	-1	0.50%	78	15.38
	0	0.11%	36	42.02
	1	0.11%	36	42.58
	2	0.05%	11	59.62
	3	0.02%	2	65.28

Table 3. Results of testing filtering in GRASP2-1 construction.

Table 3 shows that the GRASP method discards more constructed solutions for low values of λ and therefore requires less CPU time (when compared to larger λ-values). For example, with $tmax = 10$ and $\lambda = 1$, we can observe that 41 constructed solutions are discarded out of the 100 on average; which means that the improvement method is only applied to the remaining 59 solutions. The associated CPU time is therefore lower than in the regular GRASP implementation without the filtering strategy (37.74 seconds on average, which is significant lower than the 88.54 reported of the regular GRASP2-1 application). The average percentage deviation of this variant is 0.12%, instead of the 0.00% for the unfiltered GRASP2-1, which can be considered relatively good for its associated running time. On the other hand, this table shows that there are small differences when using $tmax = 10$ and when

using $tmax = 20$, with a slightly improvement in the former case.

We now undertake experiments to compare the new GRASP method, GRASP2-1, to the three previous GRASP methods in the entire set of instances. Note that the PrevGRASP2 method (Morán-Mirabal et al. 2013) was designed for the handover minimization in mobility networks, and therefore it can only be applied to the MM instances since it does not target general CCP instances. We therefore split Table 4 into two sections. The first section compares PrevGRASP1 and PrevGRASP3 to the new GRASP2-1 on the Ranreal and DB instances. The second section compares PrevGRASP2 to GRASP2-1 on the MM instances. The results in Table 4 indicate that GRASP2-1 is the best performing GRASP variant. The statistical analysis confirms it. We applied the Friedman non-parametric statistical test to the data in Table 4 and obtained a $pvalue = 0.001 < 0.01$, which indicates the existence of significant performance differences among the three methods. Additionally, we applied the Wilcoxon test between PrevGRASP3 and GRASP2-1 and obtained a $pvalue < 0.01$, indicating that there are significant performance differences between these two methods.

Procedure	$\boldsymbol{Dev}$	$\boldsymbol{Best}$	$\boldsymbol{Score}$	$\boldsymbol{Time}$
Ranreal and DB instances				
PrevGRASP1	5.67%	7%	0.38	60.00
PrevGRASP3	9.62%	27%	0.28	60.00
GRASP2-1	0.05%	70%	0.77	60.40
MM instances				
PrevGRASP2	0.80%	40%	0.40	60.00
GRASP2-1	0.79%	60%	0.60	60.00

Table 4. Comparison of GRASP methods.

We now compare our new three methods: GRASP2-1, IG1 and IG-GRASP on the entire data set, and report the results in Table 5. We include in this table a standard Simulated Annealing, SA (Johnson et al. 1989) based on the same neighborhood $N_4(x)$, as a baseline method based on randomization. Note that we could have considered other randomized metaheuristics, such as PSO or GAs (see for example De et al. 2016, De et al. 2017, or Pratap et al. 2016), which probably would provide better solutions than this straightforward SA. However, the inclusion of this method is just to evaluate how a basic randomized procedure would perform in this context.

Results in Table 5 provide an important lesson about the way in which greediness and randomization are combined in the different methods. It turns out that the IG1 approach seems more effective than the GRASP methodology to solve the CCP instances. In particular, IG1 is able to match 38% of best-known solutions while GRASP2-1 only matches 2% of them. As expected, improved outcomes are obtained when these methodologies are hybridized. In particular, IG-GRASP obtains 70% of the best-known solutions. We applied the Friedman non-parametric statistical test to the results for all instances and obtained a $pvalue < 0.01$, indicating the existence of significant performance differences among the methods. Additionally, we also applied the Wilcoxon test between IG1 and IG-GRASP obtaining a significant $pvalue = 0.005 < 0.01$, which confirms the superiority of IG-GRASP.

Procedure	*Dev*	*Best*	*Score*	*Time*
GRASP2-1	2.39%	2%	0.36	60.20
IG1	0.54%	38%	0.78	60.00
IG-GRASP	0.24%	70%	0.90	63.30
SA	5.40%	0%	0.01	60.00

Table 5. Performance comparison of new approaches.

In an attempt to understand the reasons for the superiority of IG-GRASP with respect to GRASP2-1, we perform a further analysis. Specifically, we recorded the objective function value of both the constructive phase and the improvement phase for the first 50 iterations in each method. Figure 9 shows the results. In GRASP2-1 the constructed solution is obtained from scratch at each iteration. However, in IG-GRASP the constructed solution is partially destroyed and reconstructed, which results in better outcomes than GRASP2-1. This figure clearly shows that the constructed solutions in IG-GRASP are in a higher position in the graphic than the constructed solutions in GRASP2-1. We believe that this figure provides evidence of the robustness of the IG1 approach, which consistently obtains high-quality solutions.

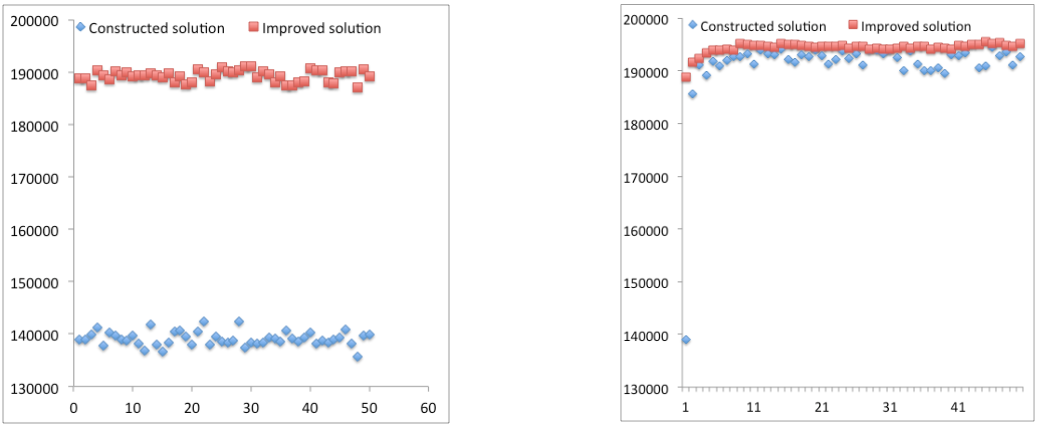

Fig 9. First 50 iterations of GRASP2-1(left) and IG-GRASP(right).

Figure 9 also supports an important point in terms of metaheuristic methodologies. The inclusion of memory structures can provide better outcomes than the memory-less methods. As a matter of fact, this figure clearly shows that the partial destruction and posterior reconstruction of a solution performs much better than the construction of a solution from scratch. In other words, it seems that even low and medium quality solutions may contain subset of elements that combined with other elements can lead to high-quality solution. In terms of memory structures, we can simply conclude that it is better to retain certain structures in the search process. An open question is if random or selective selection of these elements can make a difference. We refer the interested reader in this topic to the strategies coined under the term of "vocabulary building" introduced by Glover and Laguna (1997).

We include an additional experiment to evaluate the Matheuristic described in Section 3. In particular, we run the IG-GRASP on the 15 instances in our training set and apply the Math post-processing in which a fraction φ of the edges is fixed in the mathematical programming formulation, solved with Gurobi. Table 6 shows the average percentage deviation, $Dev.$, of the solutions obtained with different

values of φ, as well as the number of instances in which Gurobi is able to improve upon the heuristic solution, $\#Improve$, and the average running time, $Time$. We run Gurobi for a maximum of 1,800 seconds and report the total time used. This table shows that, as expected, when the φ parameter takes a low value (close to 0), the running time is relatively high, since the mathematical model is large. On the hand, when φ approaches to 1, many variables are fixed and Gurobi solves a relatively small model. In this case, computing times are very modest.

Procedure	φ	Dev	$\#$ $Improved$	$Time$
IG-GRASP	---	0.00%	0	62.00
IG-GRASP+Math	0.1	3.85%	4	1324.73
	0.25	-0.27%	5	662.99
	0.50	-0.49%	5	300.83
	0.75	-0.26%	5	72.25
	0.90	-0.25%	5	64.17

Table 6. Matheuristic performance on training set of instances.

In all the cases, the mathematical programming post-processing is only able to improve the initial heuristic solution in a small fraction of the instances tested. Note however, that these heuristic solutions are obtained with a complex metaheuristic (IG-GRASP), and could be the optimal solution of the problem. Special mention deserves the case in which φ=0.9 in which we are able to identify 5 new best solutions with a small extra running time.

5. Conclusions

In this paper, we have tackled the difficult CCP problem to investigate the crucial issue of balancing randomization and greediness in two multi-start heuristic search methods, Iterative Greedy (IG) and Greedy Randomized Search Procedure (GRASP). The GRASP methodology constructs *independent* solutions, while IG constructs *linked* solutions obtained by partially rebuilding previous ones. From an artificial intelligence perspective, they represent different approaches to problem solving. Although both are the result of the combination of greediness and randomization in the constructive process, GRASP is a **memory-less** method, while IG is a **memory-based** method. It is important to point out that memory-based designs have been extensively study in the context of improvement methods, such as the well-known tabu search methodology. However, memory-based methods has been mostly ignored in constructive algorithms. This is specially surprising since memory-based constructions were actually proposed in early tabu search designs, and developed under the name of strategic oscillation.

In this paper, we first reviewed previous GRASP designs for the CCP, and then proposed several algorithms for this problem, including a new 2-1 exchange neighborhood within the GRASP framework. Additionally, we propose an IG algorithm, and a hybrid of these two methods, called IG-GRASP with a wider consideration of the neighborhoods structures. Finally, we tested the efficiency of a filtering mechanism in the GRASP methodology. The results of our experiments using 60 benchmark instances and the associated statistical tests indicate that the hybridization IG-GRASP compares favorably to the other existing and newly proposed procedures. The investigation presented here provides an important lesson about the way in which greediness and randomization interact within the different methods considered here. It turns out that the IG1 methodology provides a more

effective framework than the GRASP methodology to solve the CCP instances. Hence, we argue that this work demonstrates that memory-based construction is an effective mechanism within multi-start heuristic search techniques.

Acknowledgments

This research has been partially supported by the University of Valencia (UV-INV-EPDI15-274064), *Ministerio de Educación Cultura y Deporte o*f Spain (Grant Ref. PRX12/00016), the *Ministerio de Economía y Competitividad* of Spain (Grant Ref. TIN2015-65460) and the Generalitat Valenciana (Prometeo 2013/049).

References

Aiex, R. M., M. G. C. Resende, and C. C. Ribeiro (2007).: "TTT plots: A PERL program to create time-to-target plots," *Optimization Letters* 1, no. 4: pp. 355-366.

Chaves, A.A. and L.A.N. Lorena (2010) Clustering search algorithm for the capacitated centered clustering problem, Computers and Operations Research, vol. 37(3), pp. 552-558.

De, A., S.K. Kumar, A. Gunasekaran, M.K. Tiwari (2017) "Sustainable maritime inventory routing problem with time window constraints", *Engineering Applications of Artificial Intelligence,* vol. 61, pp. 77-95

De, A., A. Gunasekaran, N. Subramanian, M.K. Tiwari (2016) "Composite particle algorithm for sustainable integrated dynamic ship routing and scheduling optimization", Comp. and Industrial Engineering, Vol. 96, pp. 201-215.

Deng, Y. and J.F. Bard (2011).: "A reactive GRASP with path relinking for the capacitated clustering," *Journal of Heuristics*, vol. 17, pp. 119-152.

Duarte, A., J. Sánchez-Oro, M. Resende, F. Glover, and R. Martí (2015).: "GRASP with Exterior Path Relinking for Differential Dispersion Minimization," *Information Sciences*, vol. 296, pp. 46-60.

Fan, Z. P., Y. Chen, J. Ma and S. Zeng (2011).: "A hybrid genetic algorithmic approach to the maximally diverse grouping problem," *Journal of the Operational Research Society*, vol. 62, pp. 92-99.

Fanjul-Peyroa L. and R. Ruiz (2010).: "Iterated greedy local search methods for unrelated parallel machine scheduling," *European Journal of Operational Research*, vol. 207(1), pp. 55-69.

Feo, T., O. Goldschmidt and M. Khellaf (1992).: "One-half approximation algorithms for the k-partition problem," *Operations Research*, vol. 40, pp. S170-S173.

Ferreira, C.E., A. Martin, C.C. de Souza, R. Weismantel, and L.A. Wolsey (1998).: "The node capacitated graph partitioning problem: A computational study," *Mathematical programming*, vol.81, pp. 229-256.

Festa P. and M. G. C. Resende (2009).: "An annotated bibliography of GRASP, Part I: Algorithms," *International Transactions in Operational Research*, vol. 16, pp. 1-24.

França, P.M., Sosa, N.M. and V. Pureza (1999).: An adaptive tabu search algorithm for the capacitated clustering problem, International Transactions in Operational Research, vol. 6(6), pp. 665-678.

Gallego, M., A. Duarte, M. Laguna, and R. Martí (2009).: "Hybrid heuristics for the maximum diversity problem," *Computational Optimization and Applications*, vol. 44(3), pp. 411, 2009.

Gallego, M., M. Laguna, R. Martí, and A. Duarte (2013).: "Tabu search with strategic oscillation for the maximally diverse grouping problem," *Journal of the Operational Research Society*, vol. 64, pp. 724-734.

Ghosh, J.B. (1996).: "Computational aspects of the maximum diversity problem," *Operations Research Letters,* vol. 19, pp. 175-181.

Glover, F. (1977).: "Heuristics for integer programming using surrogate constraints," *Decision Sciences,* vol. 8 (1), pp. 156-166.

Glover, F., C.C. Kuo and K.S. Dhir (1995).: "A discrete optimization model for preserving biological diversity," *Applied Mathematical Modeling,* vol. 19, pp. 696-701.

Glover, F. and M. Laguna (1997).: "*Tabu Search"*, Kluwer Academic Publisher: Boston.

Glover, F., C.C. Kuo and K.S. Dhir (1998).: "Heuristic algorithms for the maximum diversity problem," *Journal of Information and Optimization Sciences,* vol. 19 (1), pp. 109-132.

Hart J.P. and A.W. Shogan (1987).: “Semi-greedy heuristics: An empirical study,” *Operations, Research Letters*, vol. 6, pp. 107-114.

Jacobs L.W. and M.J Brusco. (1995).: “A local-search heuristic for large set-covering problems,” *Naval Research Logistics*, vol. 42(7), pp. 1129-1140.

Johnson, D. S., C. R. Aragon, L. A. McGeoch and C. Schevo (1989).: “Optimization by Simulated Annealing: An experimental Evaluation: Part I Graph Partitioning,” *Operations Research*, vol. 37 (6), pp. 865-892.

Laguna, M., and R. Martí (1999).: “GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization” INFORMS Journal on Computing 11(1), 44-52.

Lozano, M., F. Glover, C. García-Martínez, F.J. Rodríguez and R. Martí (2014).: “Tabu search with Strategic Oscillation for the Quadratic Minimum Spanning Tree,” IIE Transactions, vol. 46, pp. 414-428.

Lozano, M., D. Molina and C. García-Martínez (2011).: “Iterated greedy for the maximum diversity problem,” European Journal of Operational Research, vol. 214 (1), pp. 31-38.

Martí, R., M. Resende, C. Ribeiro (2013).: “Multi-Start Methods for Combinatorial Optimization,” *European Journal of Operational Research,* vol. 226, pp. 1-8.

Martí, R. and F. Sandoya (2013).: “The equitable dispersion problem,” *Computers and Operations Research*, vol. 40, pp. 3091-3099.

Martínez-Gavara, A., V. Campos, M. Gallego, M. Laguna and R. Martí (2015).: “Tabu Search and GRASP for the capacitated clustering problem,” *Computational Optimization and Applications*, vol. 62, pp. 589-607.

Metropolis, W., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller (1953).: “Equation of State Calculation by Fast Computing Machines,” *Journal of Chemical Physics*, vol. 21, pp. 1087-1092.

Mishra, N. , A. K. Choudhary M. K. Tiwari (2008).: “Modeling the planning and scheduling across the outsourcing supply chain : a Chaos-based fast Tabu–SA approach” *International Journal of Production Research* , vol 46, no. 13 , pp. 3683-3715.

Morán-Mirabal, L. F., J. L. González-Velarde, M. G. C. Resende, and R. M. A. Silva (2013).: Randomized heuristics for Handover minimization in mobility networks” *Journal of Heuristics*, vol. 19, pp. 845-880.

Osman, I.H., H. Christofides (1994).: Capacitated clustering problems by hybrid simulated annealing and tabu search. International Transactions in Operational Research 1 (3), pp. 317-336.

O’Brien, F. A. and J. Mingers (1995).: “The equitable partitioning problem: a heuristic algorithm applied to the allocation of university student accommodation,” Warwick Business School, *R. Paper* 187.

Pratap, S., M. Kumar, D. Saxena, M.K. Tiwari (2016), Integrated Scheduling of Rake and Stockyard Management with Ship Berthing: A Block-Based Evol. Algorithm, *Int. Journal of Prod.Research,* vol. 54(14), pp. 4182-4202.

Resende, M.G.C., R. Martí, M. Gallego and A. Duarte (2010).: “GRASP and path relinking for the maxmin diversity problem,” *Computers and Operations Research,* vol. 37(3), pp. 498- 508.

Resende, M.G.C and C.C. Ribeiro (2010).: “Greedy randomized adaptive search procedures: Advances and applications”. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, pp. 293–319. Springer,

Ribeiro, C.C., E. Uchoa, and R.F. Werneck (2002).: “A hybrid GRASP with perturbations for the Steiner problem in graphs”. *INFORMS Journal on Computing*, vol. 14, pp. 228–246.

Ruiz, R., and T. Stützle (2008).: “An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives”. *European Journal of Operational Research*, vol. 187(3), pp. 1143-1159.

Scheuerer, S. and R. Wendolsky (2006).: A scatter search heuristic for the capacitated clustering problem, European Journal of Operational Research 169 (2), 533-547.

Swarnkar, R. and M.K. Tiwari (2004).: Modelling machine loading problem of FMSs and its solution methodology using a hybrid tabu search and simulated annealing based heuristic approach, *Robotics and Computer Integrated Manufacturing*, vol. 20, pp. 199-209.

Weitz, R. R. and M. T. Jelassi (1992).: “Assigning students to groups: a multi-criteria decision support system approach,” *Decision Sciences,* vol. 23, no. 3, pp. 746-757.

Weitz, R. R. and S. Lakshminarayanan (1998).: “An empirical comparison of heuristic methods for creating maximally diverse groups,” *Journal of the Operational Research Society,* vol. 49, no. 6, pp. 635-646.

Ying, K.C. and H.M. Cheng (2010).: “Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic,” Expert Systems with Applications, vol. 37(4), pp. 2848-2852.