J Heuristics (2011) 17: 119–152
DOI 10.1007/s10732-010-9129-z

A reactive GRASP with path relinking for capacitated clustering

Yumin Deng · Jonathan F. Bard

Received: 29 November 2008 / Revised: 27 January 2010 / Accepted: 17 February 2010 /
Published online: 9 March 2010

Abstract This paper presents a greedy randomized adaptive search procedure (GRASP) coupled with path relinking (PR) to solve the problem of clustering n nodes in a graph into p clusters. The objective is to maximize the sum of the edge weights within each cluster such that the sum of the corresponding node weights does not exceed a fixed capacity. In phase I, both a heaviest weight edge (HWE) algorithm and a constrained minimum cut algorithm are used to select seeds for initializing the p clusters. Feasible solutions are obtained with the help of a self-adjusting restricted candidate list that sequentially guides the assignment of the remaining nodes. At each major GRASP iteration, the list length is randomly set based on a probability density function that is updated dynamically to reflect the solution quality realized in past iterations. In phase II, three neighborhoods, each defined by common edge and node swaps, are explored to attain local optimality. The following exploration strategies are investigated: cyclic neighborhood search, variable neighborhood descent, and randomized variable neighborhood descent (RVND). The best solutions found are stored in an elite pool.

In a post-processing step, PR is applied to the pool members to cyclically generate paths between each pair. As new solutions are uncovered, a systematic attempt is made to improve a subset of them with local search. Should a better solution be found, it is saved temporally and placed in the pool after all the pairs are investigated and the bottom member is removed. The procedure ends when no further improvement is possible. Extensive computational testing was done to evaluate the various combinations of construction and local search strategies. For instances with up to 40 nodes and 5 clusters, the reactive GRASP with PR found optimal solutions within a

Y. Deng · J.F. Bard (✉)
Graduate Program in Operations Research and Industrial Engineering, The University of Texas, Austin, TX 78712-0292, USA
e-mail: jbard@mail.utexas.edu

Y. Deng
e-mail: deng@mail.utexas.edu

negligible amount of time compared to CPLEX. In general, the HWE algorithm in the construction phase, RVND in the local search phase, and the use of PR provided the best results. The largest instances solved involved 82 nodes and 8 clusters.

Keywords GRASP · Capacitated clustering problem · Path relinking · Variable neighborhood search

1 Introduction

Clustering primarily involves the partition of objects or data points into different groups to optimize some weighted measure of distance between them. A large variety of applications exist in such areas as manufacturing, network design, pattern recognition, mail delivery, habitat classification, facility location, and statistical data analysis, to name the most prominent (e.g., see Al-Sultan and Khan 1996; Bard and Jarrah 2009; Daganzo 2005; Kaufman and Roussweuw 1990; Laporte et al. 1989). In some of these applications, the number of clusters is given while in others the objective is to find the minimum number that satisfies a set of knapsack-type constraints. In this paper, we address the constrained version of the problem and present a greedy randomized adaptive search procedure (GRASP) to find solutions. Such procedures generally have a construction phase and an improvement phase (Feo and Resende 1995; Kontoravdis and Bard 1995; Rojanasoonthon and Bard 2005). In developing the methodology, we included several options for each of these phases that markedly improved overall performance. For phase I, we designed both a heaviest weight edge algorithm and a constrained minimum cut scheme for constructing feasible solutions. For phase II, we explored the use of cyclic neighborhood search, variable neighborhood descent (VND) (Mladenovic and Hansen 1997; Hu et al. 2008), and a randomized version of the latter, to achieve local optimality. In the final step, path relinking (Glover et al. 2000) was performed on the top candidates to see if any better solutions could be uncovered on the paths between them. The design and integration of these features with a GRASP framework represents the major contribution of the research.

The specific problem that motivated the work arose from our collaboration with facility planners at mail processing and distribution centers within the US Postal Service (USPS). One of their recurrent tasks is to design zones to help rationalize the bulk movement of mail by powered industrial vehicles (PIVs). Over the course of the day, PIVs are used to transfer mail to and from the docks and between the various workcenters. Each workcenter performs a specific operation such as canceling stamps, barcoding envelops, and sorting letters to carrier routes. The pickup and drop off locations are called control points and can be regarded as fixed nodes in two-dimensional network. The problem of specifying the zones can be formulated as a mixed integer program. The difficulty in finding optimal solutions stems from its combinatorial nature. In our initial testing, we were unable to achieve convergence with CPLEX 11.0 for instances with more than 40 nodes. Therefore, we took a heuristic approach.

The construction of PIV zones falls into the general area of capacitated clustering, which is further discussed in the next section along with the related literature. In

Sect. 3, the mathematical formulation of the problem is given followed by our solution methodology which includes a reactive GRASP, two initiation procedures, our enhanced neighborhood search techniques, and path relinking. In each case, the algorithm is described and a pseudocode is given. To test the methodology, we randomly generated a large number of instances using data provided by the USPS. The results show that high quality solutions can be obtained for these instances, as well as for those solved by Mehrotra and Trick (1998). We close in Sect. 6 with an assessment of the overall approach.

2 Background and literature review

Various versions of the clustering problem have been extensively studied since the 1960s, with virtually all of them being NP-hard in the strong sense (Brucker 1978). Because the literature is vast, we tried to cite the more recent work as well as the work that relates directly to our methodology in order to avoid an explosion of the reference list. Seminal papers, if not included, can be found within the articles cited.

Mulvey and Beck (1984) proposed one of the first models for what has become known as the capacity clustering problem (CCP). In the original formulation, the objective was to find up to p capacitated clusters centered at a to-be-determined median such that the collective dissimilarity between each customer and its median is minimized. Their context was sales force territory design. In formulating the CCP, let $y_{ik} = 1$ if data point i is in cluster k and 0 otherwise, and let $z_k = 1$ if data point k is the median of cluster k and 0 otherwise ($i = 1, \ldots, n$; $k = 1, \ldots, n$). The basic model is

$$\text{Minimize} \quad \sum_{i=1}^{n}\sum_{k=1}^{n} c_{ik} y_{ik} \tag{1a}$$

$$\text{subject to} \quad \sum_{k=1}^{n} y_{ik} = 1, \quad i = 1, \ldots, n \tag{1b}$$

$$\sum_{i=1}^{n} w_i y_{ik} \le C_k z_k, \quad k = 1, \ldots, n \tag{1c}$$

$$\sum_{k=1}^{n} z_k \le p \tag{1d}$$

$$y_{ik} \in \{0, 1\}, \quad z_k \in \{0, 1\}, \quad i = 1, \ldots, n; k = 1, \ldots, n \tag{1e}$$

where w_i is the service demand of customer i, C_k is the capacity of cluster k, $p \ge \lceil \sum_{i=1}^{n} w_i / (\frac{1}{n} \sum_{k=1}^{n} C_k) \rceil$ is the number of clusters, and $c_{ik} = (\sum_{l=1}^{s} (a_{il} - a_{kl})^2)^{1/2}$ is the dissimilarity measure between i and its median k. In the expression for c_{ik}, the vector $\mathbf{a}_i \equiv (a_{i1}, \ldots, a_{is})$ represents the s attributes associated with data point i (or median k when appropriate). When points on a plane are being clustered, $\mathbf{a}_i$ is the two-dimensional vector of their X- and Y-coordinates and c_{ik} is the Euclidean norm.

Model 1 is known as the p-median capacitated clustering problem (p-CCP) when C_k is homogeneous (Ahmadi and Osman 2005; Lorena and Senne 2004).

The objective function 1a in effect minimizes the sum of the "distance" between each pair of data points in a cluster. In the formulation, all n data points are candidates for one of the p medians. Constraints 1b ensure that each data point is assigned to exactly one cluster, and constraints 1c limit the demand of each cluster k to its capacity C_k. Constraint 1d restricts the number of clusters created to p and is written as an inequality because it may not be economical to use the full capacity of the system. Logical restrictions are placed on the variables in Eq. 1e. If the redundant constraints $y_{ik} \leq z_k$ $(i, k = 1, \ldots, n)$ are added to the model, then a stronger relaxed formulation is obtained. In that case, when y_{ik} is integral z_k will be integral as well so the binary restriction on those variables can be replaced by $z_k \in [0, 1], k = 1, \ldots, n$.

A variant of model 1 known as the p-centered capacitated clustering problem (p-CCCP) arises when the median is replaced by the centroid (Negreiros and Palhano 2005). This results in a nonlinear objective function because the dissimilarity weight is now $c_{ik} = \|\mathbf{a}_i - \boldsymbol{\zeta}_k\|^2$, where $\boldsymbol{\zeta}_k \in \Re^s$ is a free variable that locates the geometric center of the cluster. In either case, a wide variety of solution strategies and techniques have been developed, from neural networks and genetic algorithms, to fuzzy sets, GRASP, and alternative c-means; e.g., see Chiou and Lan (2001), and Osman and Ahmadi (2007).

Cano et al. (2002) proposed a GRASP to solve the p-centroid uncapacitated clustering problem. Since the performance of GRASP is affected by the quality of the partial initial solution, their first step was to generate good seed candidates, which is also our first step. They then applied a probabilistic greedy Kaufman initialization in the construction phase (Kaufman and Roussweuw 1990). The Kaufman procedure identifies p dispersed points as the cluster centroids. In the improvement phase, the k-means method was used for local search. Testing was done on eight real-world benchmark data sets, the largest involving 2310 data points, 19 attributes and 7 clusters. The results showed that Kaufman-based procedure outperformed its counterparts such as random selection, Forgy's method and MacQueen's method [for a discussion of the aforementioned methods, see Hansen and Mladenovic 2001 and Kaufman and Roussweuw 1990].

Ahmadi and Osman (2005) combined GRASP and adaptive memory programming to solve the p-CCCP. The possible centers were ranked and placed on a fixed-length restricted candidate list (RCL). At each phase I iteration, one was selected randomly using a probability measure that was updated to reflect the performance of the elite (improving) solutions. The updating procedure was aimed at balancing the so-called density and intensity of the centers. A similar idea is applied in this paper except that we use the probability measure to control the RCL length rather than to select nodes in phase I (cf. Prais and Ribeiro 1999). In the improvement phase, the authors applied a restricted 1-interchange. Intensification, diversification and aspiration were also considered by setting criteria to determine whether an improved solution should be placed in the elite solution pool. Five randomly generated data sets were used to test the algorithm. The largest instances contained 150 data points and 15 clusters.

Mehrotra and Trick (1998) used column generation and a specialized bounding technique to solve the maximization version of CCP. Their pricing subproblem took the form of what they called a *maximum weight cluster problem*; a tight upper bound was obtained by solving a transportation problem. Branching was governed by the Ryan-Foster rule but it was often unnecessary to go beyond the root node due to the integrality of the linear programming solution. Testing was done on a DEC Alpha Model 300 using the same data sets as Johnson et al. (1993) who investigated a compiler design problem. The largest instance solved contained 61 nodes and 187 edges, and consumed 352 sec for a right-hand-side value in Eq. 1c of $C_k = 450$ for all k and 394 sec for $C_k = 512$.

Barreto et al. (2006) used a sequential heuristic to find solutions to the capacitated location routing problem, a combination of a facility location problem and a capacitated vehicle routing problem. The proposed algorithm first solves a clustering problem to group the customers (nodes), and then a vehicle routing problem (VRP) to obtain the routes that were subsequently improved by local search. Two kinds of algorithms (hierarchical and non-hierarchical) and six proximity metrics (single linkage, complete linkage, group average, centroid measure, ward measure, saving measure) were proposed and tested for the clustering problem. Optimality gaps of less than 5% were obtained, on average, for instances as large as 318 customers and 4 distribution centers, 150 customers and 10 distribution centers, and 117 customers and 14 distribution centers.

In the development of algorithms for the VRP, it is common to follow the logic of cluster first, route second. Newell and Daganzo (1986) approached the capacitated VRP with a single depot by grouping the customers (nodes) into zones and visiting the nodes within zones in order of their longitude coordinates. For the problems studied, the nodes were distributed randomly with a density function δ and the zones were constructed as wedge-shaped sectors elongated toward the depot. The overall objective was to minimize the expected total travel distance.

Ouyang (2007) extended the work of Newell and Daganzo (1986) by developing a systematic approach to obtain an optimal zone design. The problem studied focused on the construction of vehicle routing zones (VRZ) for given shape and size requirements, as described by Newell and Daganzo. Initially, a set of wedge-shaped zones was created satisfying these requirements. The wedges were then conformally mapped into square zones and a disk model was applied to obtain an approximately optimal partition. Further refinements were carried out by the weighted centroidal Voronoi tessellation algorithm to balance the delivery loads within the zones. From the reported computations, the proposed methodology was seen to outperform an adaptation of the Clarke-Wright heuristic with significant advantage being evidenced for large instances.

3 Mathematical formulation

For a given set of nodes V and connecting edges E, we wish to partition V into p clusters such that the sum of the "benefits" associated with the edges within each cluster is maximized and the sum of the node weights in each cluster falls with the interval $[C^{\min}, C^{\max}]$. For the PIV application with n control points, the problem can be modeled on a graph $G = (V, E)$, where $i \in V$ must appear in exactly one cluster and edge $e = (i, j) \in E$ exists in G only if there is some flow between its endpoints i and j over the week. In creating the model, we make use of the following notation.

Indices and sets

k	index for clusters
i, j	indices for nodes; $i, j \in V$
e	index for edges in G; $e \in E$

Parameters

c_e	weight of edge $e \in G$; $c_e \equiv c_{ij}$, where $i, j \in V$ such that $(i, j) = e \in E$
w_i	weight of node $i \in G$
p	number of clusters to be created
$C^{\max}$	maximum permitted weight of nodes in each cluster
$C^{\min}$	minimum required weight of nodes in each cluster

Variables

x_{ek}	1 if edge e has both its endpoints in cluster k, 0 otherwise
y_{ik}	1 if node i is included in cluster k, 0 otherwise

Model

$$\phi^{\text{IP}} = \text{Maximize} \quad \sum_{k=1}^{p} \sum_{e \in E} c_e x_{ek} \tag{2a}$$

$$\text{subject to} \quad \sum_{k=1}^{p} y_{ik} = 1, \quad \forall i \in V \tag{2b}$$

$$x_{ek} \le y_{ik}, x_{ek} \le y_{jk}, \quad \forall e = (i, j) \in E, k = 1, \ldots, p \tag{2c}$$

$$x_{ek} \ge y_{ik} + y_{jk} - 1, \quad \forall e = (i, j) \in E, k = 1, \ldots, p \tag{2d}$$

$$C^{\min} \le \sum_{i \in V} w_i y_{ik} \le C^{\max}, \quad \forall k = 1, \ldots, p \tag{2e}$$

$$x_{ek} \in \{0, 1\}, \quad y_{ik} \in \{0, 1\}, \quad \forall i \in V, e = (i, j) \in E, \quad k = 1, \ldots, p \tag{2f}$$

The objective in Eq. 2a is to maximize the sum of the edge weights within clusters, which is equivalent to minimizing the sum of the weights of edges between clusters. If the endpoints of edge e are not in the same cluster, then the corresponding weight c_e is not counted. Constraints 2b ensure that each node i is included in exactly one cluster, while constraints 2c and 2d specify that edge $e = (i, j)$ is in cluster k if and only if both endpoints i and j are in cluster k. Constraints 2e limit the total weight of the nodes in cluster k to be between $C^{\min}$ and $C^{\max}$. If the node weight $w_i = 1$ for all $i \in V$, then the summation $\sum_{i \in V} w_i y_{ik}$ is the total number of nodes assigned to cluster k. If Eq. 2e is omitted, it is optimal to create a single cluster; i.e., all the nodes and hence edges would be in one cluster. Binary restrictions are placed on all the variables in Eq. 2f.

Model 2 can be reduced by observing that constraints 2d are redundant when the objective function is taken into account and hence can be omitted. That is, when either y_{ik} or y_{jk} is 0, x_{ek} is 0, which gives a feasible solution to Eq. 2d; when both y_{ik} and y_{jk} are 1, x_{ek} will be 1 as well since the objective is to maximize the total weight.

A secondary consequence of this result is that, x_{ek} can be treated as a continuous variable in the range [0, 1]. Finally, the two-sided inequality 2e can be simplified by introducing additional slack variables $s_k, k = 1, \ldots, p$. With some algebra, we can rewrite Eq. 2e as follows:

$$\sum_{i \in V} w_i y_{ik} - s_k = C^{\min}, \quad \forall k = 1, \ldots, p \tag{2e$'$}$$

$$0 \le s_k \le C^{\max} - C^{\min}, \quad \forall k = 1, \ldots, p \tag{2e$''$}$$

where constraints 2e$''$ specify the bounds on the slack variables s_k. For other formulations of p-CCP, see Ferreira et al. (1998) or Mehrotra and Trick (1998).

4 Solution methodology

Model 2 is a 0-1 integer linear program of size $O(pn^2)$. For 60 data points and 5 clusters, this translates into a problem with approximately 18,000 variables and constraints in the worst case, which is likely to be beyond the capability of commercial solvers. Real instances are often much larger. Our experience with CPLEX 11.0 showed that some instances with $|V| = 40$ can be solved in a matter of minutes but when $|V| = 50$, runtimes exceed 10 hours. This is not surprising since the linear programming relaxation of model 2 is arbitrarily bad. By setting $x_{ek} = y_{ik} = 1/p$ for all e, i and k, the objective function value in Eq. 2a is $\sum_{e \in E} c_e$. In addition, symmetry plays havoc during branch and bound because many equivalent solutions can be obtained by exchanging the cluster numbers of any two clusters. This situation implies the existence of at least $(p-1)!$ alternative optima. Also, fixing $y_{i1} = 0$ at a particular node in the search tree has very little effect since $y_{ik}, k = 2, \ldots, p$, can still be nonzero. In addressing the issue of symmetry, Sherali and Smith (2001) proposed two disruptive strategies for a network design problem among others. We implemented both strategies in our original work but found that they increased rather than decreased CPLEX's runtimes and so were abandoned.

In light of these observations, we developed a reactive GRASP with the objective of finding high quality solutions to model 2. In phase I, good initial solutions are constructed in a greedy manner; in phase II, they are improved by local search. In a post-processing step, a subset of the phase II solutions are assembled in what is called an *elite pool* and subject to further investigation using path relinking (PR). When no better solutions can be found along the paths connecting any pair of pool members, the procedure terminates and the best available solution is output.

4.1 GRASP phase I

During construction, our first step is to initialize the p clusters. One of two approaches is used: the heaviest weight edges algorithm (HWE) or the constrained minimum cut algorithm (CMC). With HWE, we identify the p nodes with the largest weights and assign them in turn to the p clusters. The heaviest unassigned edges incident to these nodes are then sequentially assigned to the corresponding clusters along with their endpoints. The CMC approach makes use of a minimum cut algorithm to partition the graph into p clusters that satisfy the capacity lower bound $C^{\min}$. In either case, the partial solutions associated with the p clusters serve as seeds. The underlying motivation is to identify nodes and edges that are not likely to be in the same cluster in an optimal partition. After initialization, a reactive GRASP is called to construct feasible solutions. The details are given below.

HWE approach to the selection of p seeds

The HWE algorithm is illustrated in Fig. 1. At Step 1, sets and counters are initialized. At Step 2, the nodes are ordered from largest to smallest, that is, $w_{i_1} \geq w_{i_2} \geq \cdots \geq w_{i_{n-1}} \geq w_{i_n}$, and the heaviest node is assigned to cluster 1, the next heaviest to cluster 2 and so on until p clusters have been initialized or until $w_{i_s} = w_{i_n}$, where $s < p$. The objective is to disperse the heaviest nodes to different clusters in order to increase the chance of getting a feasible solution when capacity is tight. When $w_{i_s} = w_{i_{s+1}} = \cdots = w_{i_n}$, it becomes more effective to assign heavy edges rather than nodes as seeds.

At Step 3, an additional node is assigned to those clusters that have been initialized, or two nodes are assigned if the cluster is empty. In the former case, a free node that is the most heavily connected to the existing node is selected; in the latter case, the endpoint node of the heaviest free edge is assigned. At termination, each cluster will contain exactly two nodes that represent a partial initial solution to the problem. The complexity of the procedure is $O(p \cdot |V|^2)$.

```
Procedure: Phase_I_Initialize_HWE($V, E, c, w, p, C^{\min}, C^{\max}, x'$)
Input:   Set of nodes $V$, set of edges $E$, number of clusters $p$, edge weights matrix $c$,
         node weights vector $w$, and capacity bounds $C^{\min}$ and $C^{\max}$
Output:  Partial initial solution $x'$
Step 1:  $E_0 = E; V_0 = V; V_k = \emptyset, k = 1, \ldots, p; x'_{ik} = 0, \forall i \in V, k = 1, \ldots, p;$
Step 2:  $k = 1;$
         while ($k \leq p$ and $\max\{w_i, i \in V_0\} \neq \min\{w_i, i \in V_0\}$){
            $i^* \in \operatorname{argmax}\{w_i : i \in V_0\};$
            $V_k \leftarrow V_k \cup \{i^*\}; V_0 \leftarrow V_0 \backslash \{i^*\}; x'_{i^*k} = 1;$
            $k \leftarrow k + 1;$
         }
Step 3:  for ($k = 1, \ldots, p$){
            if ($|V_k| = 0$){
               $(i^*, j^*) \in \operatorname{argmax}\{c_{ij} : w_i + w_j \leq C^{\max}, (i, j) \in E_0\};$
               $V_0 \leftarrow V_0 \backslash \{i^*, j^*\}; V_k \leftarrow V_k \cup \{i^*, j^*\}; x'_{i^*k} = 1; x'_{j^*k} = 1;$
            } else {//one node already exists in cluster $V_k$
               $j^* \in \operatorname{argmax}\{c_{ij} : w_i + w_j \leq C^{\max}, i \in V_k, j \in V_0\};$
               $V_0 \leftarrow V_0 \backslash \{j^*\}; V_k \leftarrow V_k \cup \{j^*\}; x'_{j^*k} = 1;$
            }
         }
```

Fig. 1 Pseudocode for seed selection using the HWE algorithm in phase I of GRASP

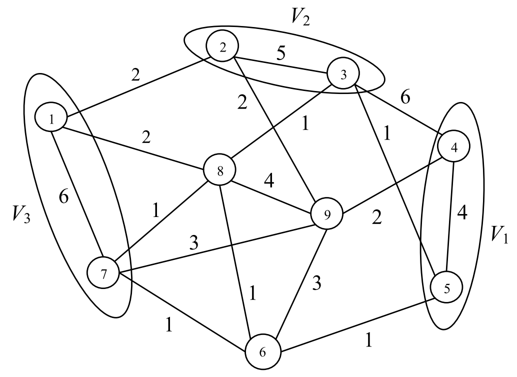

Fig. 2 Example for identifying seeds with HWE

Parameters
E_0 set of unassigned edges
V_0 set of unassigned nodes
V_k set of nodes assigned to cluster k

Figure 2 depicts the partial initial solution provided by HWE for a 9-node, 3-cluster problem with bounds $C^{\min} = 3$ and $C^{\max} = 5$. Assume that the node weights are $w_4 = 3, w_3 = 2$, and $w_i = 1, \forall i \in \{1, 2, 5, 6, 7, 8, 9\}$, and that the edge weights are as shown in the figure. Initially, $V_1 = V_2 = V_3 = \emptyset$. The heaviest node (node 4) is assigned to cluster V_1, while the second heaviest (node 3) is assigned to cluster V_2. Since the remaining nodes all have the same weight, cluster V_3 is left empty, giving $V_1 = \{4\}$, $V_2 = \{3\}$ and $V_3 = \emptyset$. In the next step, node 5 is placed into V_1 since it is the most heavily connected to node 4, and node 2 is placed into V_2 for the same reason. Finally, edge $(1, 7)$ is assigned to V_3 since it has the highest edge weight among the free edges. The algorithm ends and the initial partial solution is $V_1 = \{4, 5\}$, $V_2 = \{2, 3\}$ and $V_3 = \{1, 7\}$.

Minimum cut approach to the selection of p seeds

In this approach, we apply a constrained minimum cut scheme to partition the nodes in $G = (V, E)$ into p subsets such that the sum of the node weights in each subset $V_k, k = 1, \dots, p$, is at least $C^{\min}$, where $V = \bigcup_{k=1}^{p} V_k$ and $V_k \cap V_s = \emptyset$ for $k \neq s$. The heaviest edge in each cluster will serve as a seed while the remaining edges are removed.

The algorithm is outlined in Fig. 3. The bulk of the work is done at Step 2 with the call to CMC($V_k, E_k, C^{\min}, w, S_1, S_2$), which is a heuristic that divides G into two subsets, S_1 and S_2, such that the total weight of the edges between them is minimized while the sum of their individual node weights is at least $C^{\min}$. The problem of finding the global minimum cut in a graph is a special case of model 2, that is, when $p = 2$, $C^{\min} = 1$ in Eq. 2e, and the upper bound $C^{\max} \to +\infty$. The problem becomes NP-hard when $C^{\min} \geq 2$ and $C^{\max}$ is finite. We used Frank's (1994) polynomial-time algorithm, a slight improvement on Nagamochi and Ibaraki's (1992) algorithm, to solve the min-cut problem [its complexity is $O(|V| \cdot |E|)$] even though other lower polynomial-time algorithms exist [e.g., Karger and Stein 1996 developed a heuristic

```
Procedure: Phase_I_Initialize_CMC($V, E, c, w, p, C^{\min}, C^{\max}, x'$)
Input:  Set of nodes $V$, set of edges $E$, number of clusters $p$, edge weights matrix
        $c$, node weights vector $w$, and capacity bounds $C^{\min}$ and $C^{\max}$
Output: Partial initial solution $x'$
Step 1: $V_1 = V, V_k = \emptyset, k = 2, \dots, p; x'_{ik} = 0, \forall i \in V, k = 1, \dots, p;$
Step 2: while ($\min\{|V_k| : k = 1, \dots, p\} = 0$){
            $k^* \in \text{argmax}\{|V_k| : k = 1, \dots, p\}$;
            //Apply constrained minimum cut heuristic to $V_k$.
            call CMC($V_{k^*}, E_{k^*}, C^{\min}, w, S_1, S_2$); // see Fig. 4
            $V_{k^*} = S_1$;
            $k^* = \min\{k : |V_k| = 0, k = 1, \dots, p\}$; //pick the first empty cluster
            $V_{k^*} = S_2$;
        }
Step 3: for ($k = 1, \dots, p$){
            //Only keeps the heaviest edge in the cluster
            $(i^*, j^*) \in \text{argmax}\{c_{ij} : i \in V_k, j \in V_k\}$;
            $V_k = \emptyset; V_k \leftarrow \bigcup\{i^*, j^*\}; x'_{i^*k} = 1; x'_{j^*k} = 1;$
        }
```

Fig. 3 Pseudocode for seed selection using the CMC algorithm in phase I of GRASP

for the min-cut problem with $O(|V|^2(\log|V|)^3)$ complexity]. Our choice was based on the fact that in previous work we found Frank's algorithm easy to implement and extremely efficient on similar size graphs.

CMC works by first partitioning the subgraph $G_k = (V_k, E_k)$ into S_1 and S_2 for the unconstrained case, call it UMC, and checking each subset for feasibility. If the lower bound capacity constraint associated with, say S_1, is violated, a node is selected from S_2 and placed in S_1. At this step, the node that is most connected with the nodes in S_1, as measured by the sum of the weights of the incident edges whose endpoints are in S_1, is selected as long as the transfer does not violate the lower bound capacity constraint of S_2. The process is repeated until a feasible partition is obtained. The procedure is outlined in Fig. 4.

The same graph used to illustrate HWE will be used to illustrate Phase_I_Initialize_CMC for $p = 3$. At Step 1, $V_1 = V$ and CMC is called. At Step 1 of CMC, applying Frank's UMC algorithm to V_1 returns a minimum cut of 6 with $S_1 = \{6\}$, $S_2 = \{1, 2, 3, 4, 5, 7, 8, 9\}$ and corresponding weights $W(S_1) = 1$ and $W(S_2) = 11$. At Step 2 of CMC, we have $W(S_1) < C^{\min} = 2$ so a node must be transferred from S_2 to S_1. The calculations indicate that node 9 in S_2 is the most heavily connected to S_1 so it is selected. The updated clusters are {6, 9} and {1, 2, 3, 4, 5, 7, 8}. The operations at Step 2 stop since both S_1 and S_2 satisfy the lower bound $C^{\min}$. We put $V_1 \leftarrow S_1$, $V_2 \leftarrow S_2$ and set $S_1 = \emptyset$, $S_2 = \emptyset$. The larger set, V_2, is selected for partitioning at Step 2 of Phase_I_Initialize_CMC. Applying the UMC algorithm at Step 1 of CMC returns a minimum cut of 3 with $S_1 = \{1, 7, 8\}$ and $S_2 = \{2, 3, 4, 5\}$, both of which satisfy the lower bound constraints. Therefore, we put $V_2 \leftarrow S_1$, $V_3 \leftarrow S_2$ and set $S_1 = \emptyset$, $S_2 = \emptyset$. The operations at Step 2 of Phase_I_Initialize_CMC terminate since all three clusters are filled. The heaviest edge in each is retained and the others

```
Procedure: CMC($V_k$, $E_k$, $C^{\min}$, $c$, $w$, $S_1$, $S_2$)
Input:  Node set $V_k$, edge set $E_k$; lower bound on capacity $C^{\min}$; node weights
        vector $w$; edge weights matrix $c$
Output: Partition of nodes into subset $S_1$ and $S_2$
Step 1: Apply UMC procedure of Frank to $V_k$
        call UMC($V_k$, $E_k$, $S_1$, $S_2$);
        let $W(S) = \sum_{i \in S} w_i$;
Step 2: while ($\min\{W(S_1), W(S_2)\} < C^{\min}$){
            $k_1 = \text{argmin}\{W(S_k) : k = 1, 2\}$;
            $k_2 = \text{argmax}\{W(S_k) : k = 1, 2\}$;
            //select the most beneficial move
            $i^* = \text{argmax}_i \Big\{\sum_{j \in S_{k_1}} c_{ij}, \forall i \in S_{k_2}, W(S_{k_2}) - w_i \geq C^{\min},$
                           $W(S_{k_1}) + w_i \geq C^{\min}\Big\}$;
            Put $S_{k_2} \leftarrow S_{k_2} \backslash \{i^*\}$; $S_{k_1} \leftarrow S_{k_1} \cup \{i^*\}$;
        }
```

Fig. 4 Pseudocode of CMC scheme

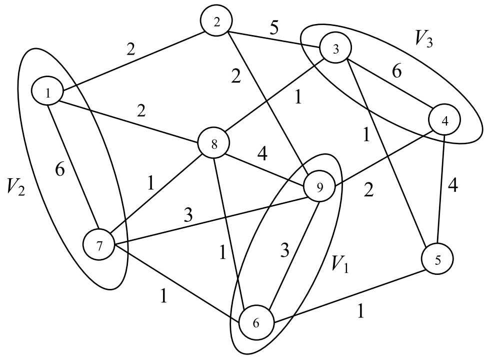

Fig. 5 Example used to illustrate CMC scheme

are removed. The final seeds for the three clusters are $V_1 = \{6, 9\}$, $V_2 = \{1, 7\}$ and $V_3 = \{3, 4\}$, as shown in Fig. 5.

Building the candidate list

Two kinds of insertions are considered when building the candidate list (CL), the structure used in GRASP to guide the construction of feasible solutions. The first corresponds to an unassigned node and the second to an unassigned edge. All feasible insertions are included in CL and sorted according to their contribution to the objective value, as measured by total edge weight that would result if the node or edge were actually added to a particular cluster. Candidates that violate the upper bound $C^{\max}$ are discarded.

Let $I(i, k)$ be the increase in the objective function value realized by inserting node i into cluster k and let $I(e, k)$ be the increase realized by inserting edge e to cluster k. Starting with the partial initial solution shown in Fig. 2 for $C^{\min} = 3$ and

Table 1 Example of CL

CL index	Edge e or node i	Cluster index k	$I(e,k)$ or $I(i,k)$
1	(8, 9)	3	10
2	(6, 9)	3	7
3	(8, 9)	2	7
4	(6, 8)	3	5
5	(6, 9)	2	5
6	8	3	3
7	9	3	3
8	9	1	2
9	9	2	2
10	(6, 8)	2	2
11	6	1	1
12	6	3	1
13	8	2	1
14	6	2	0
15	8	1	0
16	(6, 8)	1	$-\infty$
17	(6, 9)	1	$-\infty$
18	(8, 9)	1	$-\infty$

$C^{\max} = 5$, the full CL is given in Table 1. To see how these values were calculated, consider, for example, edge (8, 9). If this edge were included in cluster 3, the objective value would be 10 (that is, $c_{18} + c_{78} + c_{79} + c_{89} = 2 + 1 + 3 + 4 = 10$); if included in cluster 2, the objective value would be 7, and if included in cluster 1, the objective value would be $-\infty$ since this would lead to a violation of the upper bound $C^{\max}$. Hence, cluster 3 is the first choice for (8, 9). This insertion would increase the cluster weight from 2 to 4, which is less than $C^{\max}$.

Self-adjusting RCL

A fraction α of the top candidates in CL, up to some parameterized maximum number denoted by $\bar{l}_{\text{RCL}}$, are used to build RCL from which the next construction step is taken. The length of RCL, l_{RCL}, is determined as follows:

$$l_{\text{RCL}} = \min\{\max\{\alpha l_{\text{CL}}, 1\}, \bar{l}_{\text{RCL}}\}$$

where l_{CL} is the current length of CL. The value of α in this equation is adjusted during the GRASP iterations according to the quality of observed solutions. Prais and Ribeiro (1999) indicate that α should be within the range of (0, 1].

Let $A = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ be the finite set of possible values for α and let p_i be the corresponding probability of selecting α_i, $i = 1, \ldots, m$. Initially, p_i is uniformly distributed:

$$p_i = 1/m, \quad i = 1, \ldots, m$$

To see how these probabilities are adjusted, let ϕ^* be the best solution found in all previous GRASP iterations and let A_i be the average value of solutions obtained for $\alpha = \alpha_i$. Initially, each A_i is set to the total edge weight of the graph and 20 experiments are run by sampling α from the above uniform distribution to get 20 additional objective function values. Updating begins at this point by calculating the relative performance of the algorithm under α_i as follows:

$$q_i = \left(\frac{A_i}{\phi^*}\right)^{\delta}, \quad i = 1, \ldots, m$$

where δ is a shape parameter. For higher values of δ, q_i will be lower since $A_i \leq \phi^*$. Normalizing gives

$$p_i = q_i \Big/ \sum_{l=1}^{m} q_l, \quad i = 1, \ldots, m$$

When α_i yields relatively high average solutions A_i, it will have a high probably p_i of being selected as the iterations progress. In the implementation, we followed the suggestions of Prais and Ribeiro and set $\delta = 10, m = 10$, and $A = \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$.

Phase I initial solution construction

The partial initial solution constructed with either HWE or CMC is extended to obtain a feasible solution by sequentially adding nodes or edges to each of the p clusters. Assume that RCL is built with length l_{RCL} in accordance with above procedure. Exactly one element is randomly selected from RCL with uniformly distributed probability. The insertion corresponding to the selected element is performed to extend the current partial solution.

Again starting with the partial solution shown in Fig. 2 and with CL given in Table 1, assume that l_{RCL} is determined to be 6. The corresponding RCL is given in Table 2 and is seen to contain the top 6 candidates in CL.

If the third element is chosen, for example, then edge (8, 9) is placed in cluster 2 and the partial solution is updated. Now, CL is cleared and rebuilt along with RCL. The procedure is repeated until all nodes are assigned to one of the p clusters.

For combinatorial optimization problem like constrained clustering, it is important to note that the complexity of finding a feasible solution is the same as finding an

Table 2 Example of RCL when $l_{\text{RCL}} = 5$

RCL index	Edge e or node i	Cluster index j	$I(i, j)$ or $I(e, j)$
1	(8, 9)	3	10
2	(6, 9)	3	7
3	(8, 9)	2	7
4	(6, 8)	3	5
5	(6, 9)	2	5
6	8	3	3

optimal solution; e.g., consider the case where there is only one feasible (optimal) solution. Therefore, there is no guarantee that phase I will terminate with a feasible solution although we never encountered such a situation. Nevertheless, there are at least two options that are easily incorporated in our algorithm to deal with phase I infeasibility:

1. Increase the upper bound on the capacity constraints and then try to reduce it in phase II;
2. Add clusters one at a time and then try to eliminate them in phase II.

4.2 GRASP phase II

Three types of neighborhoods are explored in phase II. For current solution x, call them $N_1(x)$, $N_2(x)$ and $N_3(x)$, let $V_k(x)$ be the nodes in cluster k, and let $W_k(x)$ be the corresponding total node weight, $k = 1, \dots, p$. A description of the neighborhoods follows.

$N_1(x)$ (Extended node insertion) Pick a node $i \in V_k(x)$ with $W_k(x) - w_i \geq C^{\min}$. Choose a cluster $V_s(x),\ k \neq s$. If $W_s(x) + w_i \leq C^{\max}$, assign i to $V_s(x)$; otherwise, cluster s will exceed the upper bound. For the later situation, pick another node $j \in V_s(x),\ i \neq j$, and cluster $s_1 \neq s$, such that $C^{\min} \leq W_s(x) + w_i - w_j \leq C^{\max}$ and $W_{s_1}(x) + w_j \leq C^{\max}$. Shift j from $V_s(x)$ to $V_{s_1}(x)$.

$N_2(x)$ (Extended edge insertion) Pick an edge $e \in E$ with endpoints i and j. Two cases may arise; either e is in some cluster $V_k(x)$ or it spans two clusters.

(1) If $i \in V_k(x)$, $j \in V_k(x)$ and $W_k(x) - w_i - w_j \geq C^{\min}$, then find a cluster $s \neq k$ with $W_s(x) + w_i + w_j \leq C^{\max}$ and shift (i, j) from $V_k(x)$ to $V_s(x)$. If no such s exists, then go to next $e \in E$. If $W_k(x) - w_i - w_j < C^{\min}$, removing e from $V_k(x)$ would violate $C^{\min}$. In this situation stop investigating the current edge and go to next $e \in E$.

(2) If e is not an edge within a cluster, let $i \in V_{k_1}(x)$ and $j \in V_{k_2}(x)$. When $W_{k_1}(x) - w_i \geq C^{\min}$ and $W_{k_2}(x) - w_j \geq C^{\min}$, one of the following three methods is used to extend the neighborhood: (i) find a set $s \neq k_1, s \neq k_2$ with $W_s(x) + w_i + w_j \leq C^{\max}$ and shift nodes i and j to $V_s(x)$; (ii) if $W_{k_1}(x) + w_j \leq C^{\max}$, shift j from cluster k_2 to k_1; (iii) if $W_{k_2}(x) + w_i \leq C^{\max}$, shift i from cluster k_1 to k_2. If $W_{k_1}(x) - w_i < C^{\min}$ or $W_{k_2}(x) - w_j < C^{\min}$, stop and go to next $e \in E$.

$N_3(x)$ (Node exchange) For nodes $i \in V_k(x)$ and $j \in V_s(x), k \neq s$, if $C^{\min} \leq W_k(x) - w_i + w_j \leq C^{\max}$ and $C^{\min} \leq W_s(x) - w_j + w_i \leq C^{\max}$, swap i and j. Otherwise, go to next pair of nodes.

The complexity of constructing these neighborhoods is a function of $p, |V|$ and $|E|$. For each case we respectively have:

$N_1(x) \sim O(p^2 \cdot |V|^2)$
$N_2(x) \sim O(p \cdot |E|)$
$N_3(x) \sim O(\sum_{1 \leq k < s \leq p} |V_k(x)| \cdot |V_s x|)$

The corresponding pseudocodes are given in Deng (2009).

Table 3 N_1 neighborhood generated by shifting node 8

Cluster to which node 8 is moved (s)	Node to be shifted (j')	Cluster to which node is shifted (s'_1)	Total benefit gained
1	4	2	$-\infty$
	4	3	-3
	5	2	0
	5	3	0
	9	2	-1
	9	3	3
3	–	–	3

Continuing with the example in Fig. 2, assume that the current solution is $V_1 = \{4, 5, 9\}$, $V_2 = \{2, 3, 8\}$ and $V_3 = \{1, 6, 7\}$ with capacity bounds $C^{\min} = 3$ and $C^{\max} = 5$, and node weights $w_4 = 3$, $w_3 = 2$, and $w_i = 1, \forall i \in \{1, 2, 5, 6, 7, 8, 9\}$. For neighborhood N_1, the consequences of reassigning node 8 from cluster 2 to either cluster 1 or 3 are shown in Table 3. If cluster 1 is the target, then one of the nodes in cluster 1 must be removed to avoid a violation of the capacity upper bound.

For neighborhood N_2, assume that the algorithm is investigating edge $(8, 9)$ which crosses clusters 1 and 2. However, node 8 cannot be inserted into cluster 1 due to $C^{\max}$. Alternatively, if node 9 along is shifted into cluster 2, the total benefit gained will be 4 and 11 if edge $(8, 9)$ is inserted to cluster 3.

For neighborhood N_3, consider a swap between node 8 and some other node. After a simple set of calculations, we find that the best swap is between nodes 8 and 6 with benefit 1.

Capacity bounds are maintained during local search to ensure feasibility. Although it is possible to allow infeasible solutions as a strategy to overcome local optimality, such an approach would greatly increase the computational effort of phase II. In general, the GRASP philosophy is to focus the effort on phase I, not phase II [see Feo and Resende 1995 for more details]. With this in mind, diversification is introduced by accepting inferior solutions that are within some tolerance β, a parameter that is reduced dynamically by Δ after searching each of the three neighborhoods. In the basic implementation, N_1, N_2 and N_3 are explored sequentially with β starting at 1% and decreased to 0 in steps of size $\Delta = 0.2\%$. As β is reduced, the effort shifts from diversification to intensification, and when β reaches 0, no inferior solutions are accepted. A summary of phase II is given in Fig. 6. At Step 2, we cycle through the neighborhoods, terminating when no improvement is possible. We call this cyclic neighborhood search (CNS).

4.3 Basic GRASP

Given a graph $G = (V, E)$, a partial initial solution is constructed with either HWE or CMC and extended to a feasible solution using the logic surrounding RCL. Phase II is then applied a predetermined number of times to improve the current solution within the three neighborhoods. The best solution found is output as the optimum. The pseudocode for the basic reactive GRASP is given in Fig. 7 with the help of the following and aforementioned definitions.

```
Procedure: GRASP_Phase_II($x, w, c, \beta, \Delta, C^{\min}, C^{\max}, x^*$)
Input:  Current solution $x$, node weights vector $w$, edge weights matrix $c$, capacity
        bounds $C^{\min}$ and $C^{\max}$, tolerance $\beta$ and stepsize $\Delta$
Output: Local solution $x^*$ with respect to neighborhoods $N_1(x)$, $N_2(x)$ and $N_3(x)$.
Step 1: $x^* = x$;
Step 2: while ($\beta > 0$){
            Improve the current solution by local search
            call $N_1(x^*, w, c, \beta, C^{\min}, C^{\max}, x_1)$;
            call $N_2(x_1, w, c, \beta, C^{\min}, C^{\max}, x_2)$;
            call $N_3(x_2, w, c, \beta, C^{\min}, C^{\max}, x_3)$;
            $\beta = \beta - \Delta$;
            $x^* = x_3$;
        }
Step 3: $TEW(x^*) = -\infty$; $TEW(x_3) = \sum_{k=1}^{p} \sum_{i,j \in V_k(x^3)} c_{ij}$; //TEW = total edge
        weight
Step 4: while ($TEW(x_3) > TEW(x^*)$){
            $x^* = x_3$;
            call $N_1(x^*, w, c, 0, C^{\min}, C^{\max}, x_1)$;
            call $N_2(x_1, w, c, 0, C^{\min}, C^{\max}, x_2)$;
            call $N_3(x_2, w, c, 0, C^{\min}, C^{\max}, x_3)$;
            $TEW(x^*) = \sum_{k=1}^{p} \sum_{i,j \in V_k(x^*)} c_{ij}$; $TEW(x_3) = \sum_{k=1}^{p} \sum_{i,j \in V_k(x^3)} c_{ij}$
        }
```

Fig. 6 Pseudocode for Phase II of GRASP

Parameters

N^{GRASP}	number of iterations for GRASP
I^{init}	indicator of approach to build partial initial solution: $I^{\text{init}} = 0$ for HWE approach; $I^{\text{init}} = 1$ for CMC approach
iter	iteration counter

4.4 Variable neighborhood descent

VND is a systematic approach to exploring the various neighborhoods that define the local search (Mladenovic and Hansen 1997). Say there are $u_{\max}$ of them indexed by u and $N_u \subseteq N_{u+1}, \forall u = 1, 2, \ldots, u_{\max} - 1$. VND starts by searching the first neighborhood $N_1 (u = 1)$ and, in general, switches from the current neighborhood N_u to the next neighborhood N_{u+1} when N_u fails to provide an improved solution. If a better solution is obtained from N_u, then VND switches back to N_1. The procedure terminates when VND reaches the final neighborhood $N_{u_{\max}}$ and no improvement is possible. The last solution uncovered is locally optimal for all $u_{\max}$ neighborhoods.

VND has been shown to be efficient in various applications (e.g., see Hu et al. 2008). In our case, it serves as an option in phase II to increase the performance of local search even though the three neighborhoods defined above are not a subset of each other—the usual situation in which VND is applied.

```
Algorithm: GRASP
Input:   Set of nodes $V$, set of edges $E$, node weights vector $w$, edge weights ma-
         trix $c$, number of clusters $p$, capacity bounds $C^{\min}$ and $C^{\max}$, indicator $I^{\text{init}}$,
         number of iterations $N^{\text{GRASP}}$
Output:  Heuristic solution $x^{\text{best}}$
Step 1:  obtained partial initial solution $x'$
         if ($I^{\text{init}}$ equals 0) {
            call Phase_I_Initialize_HWE($V, E, c, w, p, C^{\min}, C^{\max}, x'$);
         } else {
            call Phase_I_Initialize_CMC($V, E, c, w, p, C^{\min}, C^{\max}, x'$);
         }
Step 2:  $TEW(x^{\text{best}}) = -\infty$; //$TEW$ = total edge weight
Step 3:  for ($iter = 1, \dots, N^{\text{GRASP}}$){
            construct CL and RCL, complete $x'$ to initial solution $x$ randomly;
            call GRASP_Phase_II($x, w, c, \beta, \Delta, C^{\min}, C^{\max}, x^*$);
            if ($TEW(x^{\text{best}}) < TEW(x^*)$){
               $x^{\text{best}} = x^*$;
               $TEW(x^{\text{best}}) = TEW(x^*)$;
            }
         }
```

Fig. 7 Pseudocode for basic reactive GRASP

4.5 Randomized VND

According to our initial experiments, standard VND did not lead to a balanced exploration of the three neighborhoods. Most of the effort was spent searching N_1. Even though local optimality is guaranteed, such a bias might delay convergence. To address this issue, we adopted a probabilistic weighting scheme similar to the one used for constructing RCL.

Let p_u be the probability of selecting neighborhood N_u after exploring the current neighborhood. A uniform distribution for these values is assumed initially:

$$p_u = 1/u_{\max}, \quad u = 1, \dots, u_{\max}$$

Also, let B_u be the total benefit gained by searching neighborhood N_u so far, with initial values set as follows:

$$B_u = \sum_{e \in E} c_e, \quad u = 1, \dots, u_{\max}$$

The neighborhood to be searched in the next iteration is randomly determined by the probabilities p_u. Let u^* be the current neighborhood and let b be the improvement realized from the search. The total benefit for N_{u^*} is updated by putting

$$B_{u^*} \leftarrow B_{u^*} + b$$

Note that it is possible for $b < 0$, which would indicate that a nonimproving solution was selected in the diversification step of Phase II. In that case, B_{u^*} would decrease and make N_{u^*} a less interesting option to explore. When $b > 0$, B_{u^*} will increase, suggesting that more effort should be placed on searching N_{u^*}. The probabilities p_u are hence updated to take into account the relative quality of solutions found in each neighborhood. In particular,

$$p_u = \left(B_u \middle/ \sum_{v=1}^{u_{\max}} B_v \right), \quad u = 1, \ldots, u_{\max}$$

The randomized version of VND is called RVND and is run for a predetermined number of iterations, Max_Iter. In the implementation, B is set to 1000, which is large enough to ensure that $B_u > 0, u = 1, \ldots, u_{\max}$, for the data used in the testing, and Max_Iter is set to 10.

4.6 Path relinking

During phase II, solutions that are unique are saved in a pool and sorted in descending order of their objective function values. The top N^{elite} members of the pool are selected to form the elite solution set S^{elite}. The general idea of PR is to construct a path between pairs of elements in S^{elite} to see if better solutions can be found. As described presently, feasibility is maintained at each iteration, and for a problem with p clusters, at most $p - 2$ distinct solutions will be uncovered along each path. Those that are superior to their generators are stored temporarily and, after all original pairs are examined, are inserted into S^{elite}. At the same time, the bottom elements in S^{elite} are removed to keep $|S^{\text{elite}}|$ constant. The procedure ends when the maximum number of iterations, N_{PR}, is reached or S^{elite} becomes stable, that is, the elements in S^{elite} do not change between two successive iterations.

PR was first proposed by Glover et al. (2000) and is usually combined with other metaheuristics (e.g., see Boudia et al. 2006). Given the set S^{elite} at the end of phase II, the first step is to select a pair of elements, say x_A and x_B, to serve as path generators. In this context, x_A is known as the initiating solution and x_B as the guiding solution. In attempting to construct a path that links x_A to x_B, let $V_k(x_A)$ be the node set for cluster k associated with x_A and let $V_s(x_B)$ be the node set for cluster s associated with x_B. Now, define a *similarity* measure $S(k, s)$ for $V_k(x_A)$ and $V_s(x_B)$ as follows.

$$S(k, s) = \sum_{e \in V_k(x_A) \cap V_s(x_B)} c_e$$

The value of $S(k, s)$ is the total weight of the common edges in $V_k(x_A)$ and $V_s(x_B)$. The two most similar clusters, call them k_A and s_B, associated with the elite solutions x_A and x_B, are determined by

$$(k_A, s_B) = \text{argmax}\{S(k, s) : k, s \in \{1, \ldots, p\}\}$$

where ties are broken by selecting the clusters with the smallest indices.

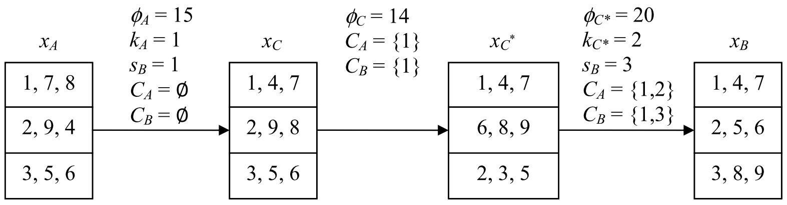

Fig. 8 An example of path generation

Given x_A and x_B, define C_A and C_B as the sets of clusters that are fixed at some iteration in the procedure. Initially, $C_A = \emptyset$ and $C_B = \emptyset$. A path from x_A to x_B is generated in the following manner. First, the most similar clusters k_A and s_B are identified according to the aforementioned logic. Cluster k_A is then modified to be exactly the same as cluster s_B by inserting and removing nodes. The cluster to which a node is moved is determined by a simple local search to minimize the decrease in objective function value. After this operation is performed a new solution x_1 emerges from x_A. The two sets C_A and C_B are updated by putting $C_A \leftarrow C_A \cup \{k_A\}$ and $C_B \leftarrow C_B \cup \{s_B\}$. Solution x_1 is then improved to be x_1^* by local search subject to the restriction that the clusters in C_A are kept constant. Now, starting from x_1^* the process is repeated to get x_2^*, and so on. Termination occurs after $p - 2$ iterations at which time all p clusters are fixed in the sets C_A and C_B; the resulting solution is exactly the same as x_B. The path generated from x_A to x_B is as follows.

$$x_A \to x_1^* \to x_2^* \to \cdots \to x_{p-2}^* \to x_B$$

Finally, let $x_{C^*} = \text{argmax}\{TEW(x_k^*) : k = 1, \ldots, p-2\}$ be the best solution found along this path. If $TEW(x_{C^*}) > \max\{TEW(x_A), TEW(x_B)\}$, then it is stored and after all pairs of elements in S^{elite} are examined, it is inserted into S^{elite}. If the capacity bounds are tight, it is possible that no feasible solution will be discovered between x_A to x_B. In that case, PR fails for x_A and x_B, and the next pair is examined.

An example of path generation based on the graph in Fig. 2 is given in Fig. 8. The nodes are to be partitioned into three clusters with $C^{\min} = 3$, $C^{\max} = 5$ and $w_4 = 3, w_3 = 2$ and $w_i = 1, i \in \{1, 2, 5, 6, 7, 8, 9\}$. Assume that x_A is $\{\{1, 7, 8\}$, $\{2, 9, 4\}$, $\{3, 5, 6\}\}$ with objective function value $\phi_A = 15$ and that x_B is $\{\{1, 4, 7\}$, $\{2, 5, 6\}$, $\{3, 8, 9\}\}$ with $\phi_B = 12$. Starting with $C_A = \emptyset$ and $C_B = \emptyset$, the goal is to generate a path from x_A to x_B. At Step 1 the clusters most similar with respect to solutions x_A and x_B are $k_A = 1$ and $s_B = 1$ with $S(k_A, s_B) = 6$. Node 8 in $V_1(x_A)$ is removed and inserted into $V_2(x_A)$ while node 4 in $V_2(x_A)$ is shifted to$V_1(x_A)$. Call the transformed solution x_C, and note that cluster 1 in x_C is exactly the same as cluster 1 in x_B; that is, $V_1(x_C) = V_1(x_B)$.

Next, the constant sets are updated giving $C_A = \{1\}$ and $C_B = \{1\}$, and a local search is performed on x_C, which results in an improved solution x_{C^*}. At the next step, the most similar clusters with respect to x_{C^*} and x_B are determined to be $k_{C^*} = 2$ and $s_B = 3$. The sets C_A and C_B are now $\{1, 2\}$ and $\{1, 3\}$, respectively. To make cluster 2 in x_{C^*} the same as cluster 3 in x_B, node 6 is selected and placed in $V_3(x_{C^*})$ while node 3 is shifted to $V_2(x_{C^*})$. The resulting solution is exactly the same as x_B.

```
Procedure: Path_generation($w, c, C^{\min}, C^{\max}, n^{PR}, x_A, x_B, x^*$)
Input:  Node weights vector $w$, edge weights matrix $c$, capacity bounds $C^{\min}$ and
        $C^{\max}$, PLS parameter $n^{PR}$, initiating solution $x_A$, guiding solution $x_B$
Output: Best solution $x^*$ found along the path from $x_A$ to $x_B$
Step 1: $C_A = \emptyset$; $C_B = \emptyset$; $TEW^* = -\infty$; $r = 1$;
Step 2: while ($x_A$ is not the same as $x_B$){
            $(k_A, s_B) = \text{argmax}\{S(k,s) : k, s \in \{1, \dots, p\}\}$;
            make $V_{k_A}(x_A)$ the same as $V_{s_B}(x_B)$ by inserting and removing nodes so
            that $x_A$ becomes $x_r$;
            // keeping the clusters $k \in C_A$ constant, if the PLS condition is satisfied
            then apply local search to $x_r$ in an attempt to obtain a better solution $x_r^*$
            if ($r$ mod $n^{PR} + 1$ equals 0) {
                call $N_1(x_r, w, c, 0, C^{\min}, C^{\max}, x)$; $x_r = x$;
                call $N_2(x_r, w, c, 0, C^{\min}, C^{\max}, x)$; $x_r = x$;
                call $N_3(x_r, w, c, 0, C^{\min}, C^{\max}, x_r^*)$;
            }
            if ($TEW(x_r^*) > TEW^*$){
                $TEW^* = TEW(x_r^*)$; $x^* = x_r^*$;
            }
            $x_A = x_r^*$; $r \leftarrow r + 1$;
        }
```

Fig. 9 Pseudocode for path generation

The best solution found along the path is x_{C^*} with $\phi_{C^*} = TEW = 20$. Since $\phi_{C^*} > \phi_A$ and $\phi_{C^*} > \phi_B$, x_{C^*} is outputted and stored for possible insertion into S^{elite}. The pseudocode for path generation is shown in Fig. 9. In our implementation, $|S^{\text{elite}}|$ is set to 20. For each pair of solutions $x_A \in S^{\text{elite}}$ and $x_B \in S^{\text{elite}}$, two paths are generated, the first starting from x_A and approaching x_B and the second taking the reverse course. For a given S^{elite}, the total number of paths is $O(|S^{\text{elite}}|^2)$. When S^{elite} becomes stable the best solution found up to that point is output.

A potentially inefficient aspect of PR is the application of local search to each solution encountered along a path. Empirically, we found that a complete local search (CLS) strategy may affect the solution quality only locally within the same basin of attraction. In addition, the current solution may have been uncovered previously so applying local search a second time is wasteful. One way to reduce the computational effort is to apply local search only after encountering n^{PR} solutions along a path, where n^{PR} is a parameter adjusted according to the solution quality. This strategy is referred to as *partial local search* (PLS) to distinguish from CLS. For p clusters, $n^{PR} \in \{1, 2, \dots, p-2\}$, where $n^{PR} = 1$ indicates that local search is applied to each solution in the path, $n^{PR} = 2$ indicates that it is applied to every second solution, and so on. Note that there are at most p solutions along a path including the initiating solution and the guiding solution. Because both of these are already locally optimal, $n^{PR} \le p - 2$.

The value of n^{PR} is randomly selected at the beginning of each path. The probability function used for this purpose is based on a performance measure $P_i(n^{PR})$, which is defined as the average objective function value over a path when the frequency of applying local search was n^{PR}; that is,

$$P_i(n^{\mathrm{PR}}) = \sum_{j=1}^{N_i} I\{j \bmod n^{\mathrm{PR}}\}A_{ij} \Big/ \sum_{j=1}^{N_i} I\{j \bmod n^{\mathrm{PR}}\}$$

where i is the index for path, N_i is the number of solutions discovered along path i, j is the index for the solutions discovered on a path, A_{ij} is the objective value of the jth solution discovered on path i and $I\{j \bmod n^{\mathrm{PR}}\}$ is a Boolean indicator function equal to 1 when $(j \bmod n^{\mathrm{PR}}) = 0$ (i.e., $\langle$true$\rangle$) and 0 otherwise. The summation $\sum_{j=1}^{N_i} I\{j \bmod n^{\mathrm{PR}}\}$ counts the total number of times local search is applied while exploring path i. If n^{PR} is not selected for path i, then $P_i(n^{\mathrm{PR}}) = 0$.

Next, we compute the accumulated performance, denoted by $AP(n^{\mathrm{PR}})$, for a particular value of n^{PR} by summing over all the paths already generated.

$$AP(n^{\mathrm{PR}}) = \sum_{i=1}^{n^{\mathrm{cur}}} P_i(n^{\mathrm{PR}})$$

Here, n^{cur} is the number of paths that have been explored up to and including the current path.

At the beginning of PR, the probability $p(n^{\mathrm{PR}})$ of selecting a particular value of $n^{\mathrm{PR}} \in \{1, 2, \dots, p-2\}$ is assigned a uniform distribution; that is,

$$p(n^{\mathrm{PR}}) = 1/(p-2)$$

After exploring a path, this function is updated as follows.

$$p(n^{\mathrm{PR}}) = AP(n^{\mathrm{PR}}) \Big/ \sum_{n^{\mathrm{PR}}=1}^{p-2} AP(n^{\mathrm{PR}})$$

Thus, values of n^{PR} corresponding to higher accumulated performance will have a higher probability of being selected.

5 Computational results

The proposed methodology was implemented in C++ and run under Ubuntu Linux on a Dell Poweredge 2950 workstation with 2 dual core hyperthreading 3.73 GHz Xeon processors and 8 GB memory. In the testing, model 2 was solved, both heuristically with the reactive GRASP and directly with CPLEX 11.0 when possible. A comparison of the results gives insight in the quality of the GRASP solutions as well as the limits of CPLEX.

The following settings were used for the GRASP.

- Both HWE and CMC schemes were applied in Phase I to construct partial initial solutions but in separate runs to allow for comparison.

- Initial value of diversification parameter: $\beta = 0.01$ with $\Delta = 0.002$ in Phase II.
- Three options were examined for local search: (1) CNS; (2) VND; (3) RVND.
- Number of GRASP iterations: $N^{\text{GRASP}} = 5 \times n^{\text{test}}$ with n^{test} being the number of nodes in the test instance.
- In PR, the maximum number of iterations is $N_{\text{PR}} = 50$, the number of elite solutions maintained is $|S^{\text{elite}}| = 20$.

After some experimentation, the following settings were used for CPLEX.

- Cut generators off.
- Emphasis of feasibility over optimality.
- Optimality tolerance EpOpt = 1E–04.
- Default frequency for MIP heuristics.

In the experiments, the methodology was tested on three data sets. The first contained relatively small instances that were randomly generated based on data provided by the USPS. Each instance was generated from a different seed. The second contained instances that reflected the full USPS problem. The third were obtained from Mehrotra and Trick (1998). In the next section, we outline the USPS application and describe how the node and edge weights were specified.

5.1 USPS application related to clustering control points

The cost of running a mail processing and distribution center (P&DC) is determined in part by the size and composition of the workforce. One of management's goals is to use as few powered industrial vehicles (PIVs) or drivers as possible to move the mail between workcenters, so restricting the number of control points (workcenters) that a driver can service would be suboptimal. However, to facilitate supervision and to avoid violating union rules, control points are first clustered into zones and then the minimum number of PIVs required to service each zone is determined. In the clustering step, it is necessary to take into account such factors as distance between nodes and transfer frequencies. Two nodes are likely to be grouped together if they are directly linked in the process flow, are relatively close to each other, and one is a frequent terminal point of the other.

In the clustering model, it is necessary to specify a measure that numerically captures these characteristics. Such a measure can be viewed as the edge weights, c_{ij}, connecting pairs of nodes i and j in a directional graph. For the test cases, we used the following formula to determine these weights.

$$c_{ij} = \frac{f_{ij} + f_{ji}}{d_{ij}} \times d_{\max}$$

The parameter f_{ij} in this equation denotes the frequency of travel from node i to node j over the planning horizon The numerator represents the traffic intensity, the denominator, d_{ij}, the length of edge (i, j), and the parameter $d_{\max}$ the maximum edge length in the graph, that is, $d_{\max} = \max\{d_{ij} : (i, j) \in E\}$. This value is used to normalize the distance d_{ij}. If the demand between two nodes i and j is high and they are close together, then the corresponding edge weight will have a relatively high value so the two nodes would likely be in the same optimal partition.

P&DCs typically have between 80 and 90 control points, each of equal weight from management's point of view, so we set $w_i = 1, \forall i \in V$. In the planning stage, the number of clusters is specified by the facility manager taking into account the daily volume, the building's footprint, the equipment layout, and the various components of the material handling system. In addition to PIVs, which consist of tugs and forklifts, facilities use fixed conveyers, rolling carts, and an assortment of other mechanisms for material handling. Mathematically, the problem is equivalent to model 2.

5.2 Random test instances

Instances of practical size cannot be solved optimally with commercial codes so to test our methodology, we randomly generated a series of data sets based on the characteristics of the Chicago P&DC. This involved the following steps.

(1) Let V be the set of control points in the original P&DC data set and let E be the corresponding set of edges. Define the density γ of the underlying graph as

$$\gamma = |E|/|E_\mathrm{C}|$$

where $|E_\mathrm{C}|$ is the number of edges when the graph is completely connected. For the given data, the density γ is approximately 0.1626. Also, let $c_{\max}$ and $c_{\min}$ be the maximum and minimum edge weights, respectively; that is, $c_{\max} = \max\{c_{ij}, (i, j) \in E\}$ and $c_{\min} = \min\{c_{ij}, (i, j) \in E\}$.

(2) Randomly select n^{test} nodes from the original P&DC data set. Let V^{test} be the corresponding set of nodes and fix $w_i = 1, \forall i \in V^{\text{test}}$. The number of edges in the completely connected test graph is $|E_\mathrm{C}^{\text{test}}| = n^{\text{test}}(n^{\text{test}} - 1)/2$.

(3) Define $m = (2 \cdot \gamma \cdot |E_\mathrm{C}^{\text{test}}|)/n^{\text{test}}$. The product $\gamma \cdot |E_\mathrm{C}^{\text{test}}|$ is the number of edges in the test graph that should be generated to maintain the same density, and m is the average number of edges incident to each node in the test graph. In our procedure we aim for $\lfloor m \rfloor$ rather than m incident edges. For example, if $n^{\text{test}} = 25$ and $\gamma = 0.1626$, then $|E_\mathrm{C}^{\text{test}}| = 300$ and $\lfloor m \rfloor = \lfloor 3.9 \rfloor = 3$, which means that on average each node is connected to 3 other nodes.

(4) Let E^{test} be the edge set of the test instance, where initially, $E^{\text{test}} = \emptyset$. For each node $i \in V^{\text{test}}$, let N_i be the set of nodes already connected to i and to begin, set $N_i = \emptyset$. If $|N_i| \geq \lfloor m \rfloor - 1$, go to next node i in V^{test}. Otherwise, let $p_j, \forall j \in V^{\text{test}} \backslash N_i, j \neq i$, be the probability for a node j to be connected to node i. This probability is computed as the ratio of the remaining number of nodes to be connected to i divided by the number of unassigned nodes: $p_j = (\lfloor m \rfloor - |N_i|)/(n^{\text{test}} - |N_i| - 1)$. If j is selected, $N_i \leftarrow N_i \cup \{j\}$, $E^{\text{test}} \leftarrow E^{\text{test}} \cup \{(i, j)\}$. The edge weight c_{ij} is uniformly generated from the interval $[c_{\min}, c_{\max}]$.

5.3 Comparison of GRASP and PR with CPLEX

In the first experiments, we compared the reactive GRASP to CPLEX using the different phase I and phase II options for instances with $n^{\text{test}} = 30$ nodes and $p = 5$ clusters. Ten instances were randomly generated in accordance with the above scheme with bounds $C^{\min} = 5$ and $C^{\max} = 8$. The model was built with CPLEX 11.0 Concert Technology version 25 and contained 2330 variables and 4386 constraints. The number of GRASP iterations was set to $N^{\text{GRASP}} = 5 \times n^{\text{test}} = 150$, and was followed by PR in all cases with either CLS or PLS. Finally, a 3600 sec time limit was placed on all CPLEX runs.

The results are summarized in Table 4. The second column lists the density of the realized graph, $\gamma^{\text{test}} = |E^{\text{test}}|/|E_{\text{C}}^{\text{test}}|$. The third and fourth columns give the results for the two combinations (HWE, CNS) and (CMC, CNS) prior to path relinking. The upper row values report the best solutions found by GRASP for the corresponding pair. The lower values in parentheses report the iteration number at which the best solutions were first discovered. Columns 5 and 6 give equivalent results for (HWE, VND) and (CMC, VND), while columns 7 and 8 report the results for (HWE, RVND) and (CMC, RVND). With the exception of problem no. 3 for combination (CMC, RVND), GRASP found identical solutions with the various phase I and phase II options. Average runtimes, t_{avg}, over the six scenarios are given in column 9.

The results for PR with CLS averaged over the six scenarios are contained in columns 10 and 11. Similar results for PR with PLS are contained in columns 12 and 13. The entries in the columns labeled "PR + CLS" were determined by averaging the results for the six scenarios in each row after PR with CLS was run. Similarly, for PR + PLS in columns 12 and 13. The t_{avg} results are for PR alone.

As can be seen in the table, PLS achieves the same solutions as CLS but in less time with the exception of problem no. 8. Other than for problem no. 3, PR could not improve the GRASP solutions since they are optimal. This is confirmed by the results from CPLEX given in columns 14 and 15. The last column provides the gap between the average PR + CLS solution and the CPLEX solution, $[(\phi^{\text{PR+CLS}} - \phi^{\text{CPLEX}})/\phi^{\text{CPLEX}}] \times 100\%$, which is zero for all cases.

Table 5, which is derived from Table 4, compares the average performance of the six phase I–phase II combinations. For $j = 1, \ldots, 6$, let $j = 1$ indicate (HWE, CNS), $j = 2$ indicate (HWE, VND), $j = 3$ indicate (HWE, RVND), $j = 4$ indicate (CMC, CNS), $j = 5$ indicate (CMC, VND), and $j = 6$ indicate (CMC, RVND). Define the average error e_j for combination j as follows

$$e_j = \frac{1}{N^{\text{test}}} \sum_{i=1}^{N^{\text{test}}} \left[\left(\phi_{ij}^{\text{PR+CLS}} - \phi_{ij}^{\text{GRASP}} \right) / \phi_{ij}^{\text{PR+CLS}} \times 100\% \right], \quad \forall j = 1, \ldots, 6$$

where $N^{\text{test}} = 10$ is the number of instances, ϕ_{ij}^{GRASP} is the best solution found by GRASP with combination j for instance i. The average error e_j measures the improvement attained by PR + CLS over the GRASP solutions in all instances for combination j. The higher e_j, the more improvement provided by PR + CLS.

The second column in Table 5 shows the average number of iterations needed to obtain the best solution for (HWE, CNS) and (CMC, CNS), respectively. The values were calculated by averaging the number of iterations inside parentheses in column three for (HWE, CNS) and column four for (CMC, CNS) in Table 4. The third column gives the average errors of (HWE, CNS) and (CMC, CNS), respectively. The next columns report the same statistics for the remaining combinations. As mentioned,

Table 4 Computational results from GRASP and CPLEX for $n^{\text{test}} = 30$, $p = 5$, $C^{\min} = 5$ and $C^{\max} = 8$

Prob	γ^{test}	GRASP											CPLEX		PR-CX
no.		GRASP solution							PR solution				Best	Time	gap
		CNS		VND		RVND		t_{avg}	PR + CLS	t_{avg}	PR + PLS	t_{avg}	solution	(sec)	(%)
		HWE	CMC	HWE	CMC	HWE	CMC	(sec)	(avg.)	(sec)	(avg.)	(sec)	found		
1	0.1908	618.97 (1)	618.97 (12)	618.97 (7)	618.97 (12)	618.97 (12)	618.97 (11)	1	618.97	5	618.97	3	618.97	10	0.00
2	0.1977	691.49 (27)	691.49 (2)	691.49 (26)	691.49 (15)	691.49 (16)	691.49 (19)	1	691.49	4	691.49	3	691.49	35	0.00
3	0.1747	667.09 (18)	667.09 (3)	667.09 (18)	667.09 (3)	667.09 (3)	665.48 (7)	1	667.09	4	667.09	3	667.09	21	0.00
4	0.1839	680.54 (6)	680.54 (12)	680.54 (37)	680.54 (17)	680.54 (1)	680.54 (14)	1	680.54	4	680.54	4	680.54	13	0.00
5	0.1609	569.67 (3)	569.67 (8)	569.67 (3)	569.67 (1)	569.67 (34)	569.67 (6)	1	569.67	7	569.67	2	569.67	18	0.00
6	0.1793	642.67 (5)	642.67 (1)	642.67 (3)	642.67 (8)	642.67 (4)	642.67 (1)	1	642.67	4	642.67	1	642.67	29	0.00
7	0.1678	618.25 (5)	618.25 (1)	618.25 (15)	618.25 (11)	618.25 (22)	618.25 (8)	1	618.25	3	618.25	3	618.25	6	0.00
8	0.1609	628.91 (1)	628.91 (19)	628.91 (3)	628.91 (10)	628.91 (4)	628.91 (2)	1	628.91	2	628.91	4	628.91	7	0.00
9	0.1540	544.62 (38)	544.62 (13)	544.62 (52)	544.62 (74)	544.62 (21)	544.62 (2)	1	544.62	3	544.62	1	544.62	5	0.00
10	0.1609	598.66 (3)	598.66 (14)	598.66 (6)	598.66 (5)	598.66 (4)	598.66 (2)	1	598.66	4	598.66	2	598.66	4	0.00

Table 5 Average performance for different phase I and phase II combinations for $n^{\text{test}} = 30$, $p = 5$, $C^{\min} = 5$ and $C^{\max} = 8$ with PR + CLS

Phase I	Phase II					
	CNS		VND		RVND	
	Avg. number of iterations	Avg. error e_j (%)	Avg. number of iterations	Avg. error e_j (%)	Avg. number of iterations	Avg. error e_j (%)
HWE	10.7	0.00	17.0	0.00	12.1	0.00
CMC	8.5	0.00	15.6	0.00	7.2	0.02

GRASP found the optimal solutions with $e_j = 0.00, \forall j = 1, \ldots, 5$ except for the last combination (CMC, RVND), $e_6 = 0.02$.

Applying HWE in phase I required a greater number of iterations on average than CMC to find the best solutions no matter which option was applied in phase II. When the phase I option was fixed, VND required more iterations than its two counterparts, which performed equally well.

The second set of initial experiments was performed on a 40-node graph for $p = 5$ clusters with bounds $C^{\min} = 5$ and $C^{\max} = 9$. Model 2 contained 4105 variables and 7846 constraints for each of the 10 instances investigated, and $N^{\text{GRASP}} = 200$ iterations. Once again, all instances were generated randomly from the USPS data.

The results are summarized in Table 6. All the GRASP–PR runs consumed much less time than CPLEX as can be seen in columns 9, 11 and 13. For problem nos. 1, 3, 4, 5, 7 and 8, CPLEX converged to the optimum; for the remaining instances the 1-hour time limit was reached before optimality could be confirmed. For the GRASP, PR improved the phase II solutions in some cases, especially when VND was applied. In all cases, CLS and PLS achieved identical solutions but PLS required slightly less time. In addition, GRASP with PR invariably provided equivalent or better solutions than CPLEX in much less time.

The average performance of the Phase I–Phase II combinations with PR + CLS for the 40-node instances is reported in Table 7. For a given Phase I option, RVND required the least number of iterations, followed by CNS and then VND. In addition, the average error was highest for VND, while the errors for CNS and RVND were roughly the same. When either CNS or VND was applied in Phase II there was little difference with respect to HWE and CMC. However, when RVND was applied, CMC required 37% fewer iterations than HWE on average.

The third set of initial experiments was performed on a 50-node graph with $p = 5$, $C^{\min} = 5$ and $C^{\max} = 12$. The optimization model contained 6380 variables and 12306 constraints, while the number of GRASP iterations $N^{\text{GRASP}} = 250$. Again, ten instances were randomly generated from the original USPS data following the aforementioned scheme.

The results are reported in Table 8. All of the GRASP runs finished within 7 sec while the PR runs finished within 30 sec. In all cases, the PR solutions were better than those provided by CPLEX except for problem no. 5 where they were identical. Note that CPLEX was never able to converge but always found feasible solutions within the allotted time.

Table 6 Computational results from GRASP and CPLEX for $n^{\text{test}} = 40$, $p = 5$, $C^{\min} = 5$ and $C^{\max} = 9$

Prob no.	γ^{test}	GRASP											CPLEX		PR-CX gap (%)
		GRASP solution							PR solution				Best solution found	Time (sec)	
		CNS		VND		RVND		t_{avg} (sec)	PR + CLS (avg.)	t_{avg} (sec)	PR + PLS (avg.)	t_{avg} (sec)			
		HWE	CMC	HWE	CMC	HWE	CMC								
1	0.1782	1043.77 (2)	1043.77 (13)	1043.77 (41)	1043.77 (4)	1043.77 (17)	1043.77 (15)	1	1043.77	10	1043.77	9	1043.77	693	0.00
2	0.2013	1127.87 (72)	1127.83 (15)	1127.87 (16)	1127.87 (1)	1127.87 (11)	1127.87 (45)	1	1127.87	10	1127.87	7	1127.87	3600	0.00
3	0.1923	1169.84 (14)	1169.84 (29)	1169.84 (182)	1158.21 (54)	1169.84 (44)	1169.84 (8)	1	1169.84	16	1169.84	7	1169.84	3260	0.00
4	0.1910	1148.93 (73)	1148.93 (42)	1148.93 (144)	1148.93 (155)	1148.93 (121)	1148.93 (20)	1	1148.93	9	1148.93	7	1148.93	1073	0.00
5	0.2026	1124.71 (104)	1125.80 (128)	1125.80 (46)	1125.80 (52)	1125.80 (10)	1125.80 (47)	3	1125.80	11	1125.80	8	1125.80	518	0.00
6	0.2090	1154.98 (77)	1158.93 (60)	1158.93 (134)	1158.93 (73)	1158.93 (145)	1158.93 (37)	4	1158.93	13	1158.93	11	1154.98	3600	0.34
7	0.1872	1065.31 (7)	1065.31 (61)	1059.64 (120)	1062.86 (191)	1065.31 (12)	1064.30 (23)	1	1065.31	9	1065.31	9	1065.31	1420	0.00
8	0.1846	1166.05 (11)	1166.05 (41)	1166.05 (11)	1166.05 (79)	1166.05 (2)	1166.05 (31)	1	1166.05	13	1166.05	9	1166.05	702	0.00
9	0.1833	1096.67 (44)	1096.67 (13)	1096.67 (36)	1096.67 (16)	1096.67 (6)	1096.67 (15)	1	1096.67	7	1096.67	5	1096.67	3600	0.00
10	0.1846	1150.02 (6)	1150.02 (30)	1150.02 (6)	1150.02 (9)	1150.02 (14)	1150.02 (7)	1	1150.02	10	1150.02	8	1150.02	3600	0.00

Table 7 Average performance for different phase I and phase II combinations for $n^{\text{test}} = 40$, $p = 5$, $C^{\min} = 5$ and $C^{\max} = 9$ with PR + CLS

Phase I	Phase II					
	CNS		VND		RVND	
	Avg. number of iterations	Avg. error e_j (%)	Avg. number of iterations	Avg. error e_j (%)	Avg. number of iterations	Avg. error e_j (%)
HWE	41.0	0.01	73.6	0.05	38.2	0.00
CMC	43.2	0.00	63.4	0.12	24.8	0.01

The average performance of GRASP is reported in Table 9 for the 50-node instances. The average error $e_j > 0$, $j = 1, \ldots, 6$, which means that on average PR + CLS found improved solutions for all combinations. When the Phase I option was fixed, VND had the highest error, followed by CNS and RVND. For CNS and VND in Phase II, the errors from HWE and CMC were nearly identical. When RVND was applied, the error from HWE was less than half of the error from CMC. With respect to the average number of iterations, VND and RVND performed equally well, while CNS required the least number of iterations no matter which option was used in Phase I.

In the fourth set of initial experiments we investigated the performance of GRASP and PR + PLS as the number of clusters p was varied from 2 to 10 for the same 30-node graph associated with problem no. 1 in Table 1. The bounds were set to be $C^{\min} = 2$ and $C^{\max} = 15$ to reduce their effect on the computations. The results were similar to those already discussed. In all cases, the optimal solution was found by GRASP and CPLEX, but runtimes differed markedly. GRASP with PR + PLS were quite stable no matter which combination was used but CPLEX had increasing difficulty as the number of clusters increased.

In the fifth set of initial experiments, a parametric analysis was performed on the bounds for $p = 5$ fixed. The bounds $[C^{\min}, C^{\max}]$ were initially set to [2, 10] and then modified in even steps to reach [6, 6]. For all runs, GRASP with PR was able to find the same optimum obtained by CPLEX but in considerably less time. Similar to the fourth set of experiments, the performance of GRASP was insensitive to the bounds while CPLEX had more difficulty as the range shrank. The full set of results for the latter two sets of experiments can be found in Deng (2009).

5.4 Application of GRASP and PR to the complete USPS dataset

In the second set of tests, we applied GRASP with PR using PLS to the full USPS dataset, which has 82 nodes and 540 edges (i.e., $|V^{\text{test}}| = 82$ and $|E^{\text{test}}| = 540$). The bounds were set as follows: $C^{\min} = 10$ and $C^{\max} = 20$. The goal was to investigate the six combinations of phase I and phase II options for a range of p values. In each run, the number of GRASP iterations $N^{\text{GRASP}} = 410$.

The results are reported in Table 10 for $p \in \{5, 6, 7, 8\}$. The first column gives the number of clusters p. The second column indicates the options used for GRASP. The third column, ϕ^{best}, is the best solution found by GRASP and PR. The fourth column indicates the improvement achieved by PR. In column 5, the value i^{best} denotes the

Table 8 Computational results from GRASP and CPLEX for $n^{\text{test}} = 50$, $p = 5$, $C^{\min} = 5$ and $C^{\max} = 12$

Prob	γ^{test}	GRASP											CPLEX		PR-CX
no.		GRASP solution							PR solution				Best	Time	gap
		CNS		VND		RVND		t_{avg}	PR + CLS	t_{avg}	PR + PLS	t_{avg}	solution	(sec)	(%)
		HWE	CMC	HWE	CMC	HWE	CMC	(sec)	(avg.)	(sec)	(avg.)	(sec)	found		
1	0.1845	1639.22	1634.33	1631.77	1633.35	1639.22	1639.22	1	1639.22	20	1639.22	13	1602.56	3600	2.29
		(59)	(5)	(244)	(155)	(39)	(30)								
2	0.1869	1685.68	1685.68	1685.68	1685.68	1685.68	1685.68	5	1685.68	11	1685.68	11	1646.34	3600	2.39
		(176)	(98)	(27)	(40)	(139)	(11)								
3	0.1714	1560.56	1550.17	1549.55	1550.17	1560.56	1556.19	5	1560.56	25	1559.38	13	1491.93	3600	4.60
		(155)	(80)	(143)	(37)	(43)	(238)								
4	0.1706	1677.50	1679.90	1678.02	1678.02	1679.90	1679.90	7	1679.90	29	1679.90	15	1647.10	3600	1.99
		(4)	(29)	(20)	(125)	(67)	(7)								
5	0.1665	1637.83	1640.17	1640.17	1640.17	1640.17	1640.17	7	1640.17	23	1640.17	15	1640.17	3600	0.00
		(4)	(90)	(32)	(180)	(67)	(198)								
6	0.1837	1658.19	1662.09	1658.19	1658.19	1662.09	1658.43	2	1662.09	20	1659.57	15	1615.09	3600	2.91
		(55)	(203)	(148)	(37)	(212)	(246)								
7	0.1796	1670.81	1670.81	1669.95	1670.81	1670.81	1670.81	1	1670.81	12	1670.81	12	1624.56	3600	2.85
		(47)	(182)	(161)	(103)	(56)	(169)								
8	0.1682	1640.90	1646.57	1644.53	1646.57	1646.57	1646.57	7	1651.09	25	1651.09	13	1594.65	3600	3.54
		(57)	(104)	(63)	(160)	(206)	(144)								
9	0.1641	1648.39	1648.39	1648.39	1648.39	1648.39	1648.39	1	1648.39	11	1648.39	12	1643.22	3600	0.31
		(12)	(66)	(175)	(204)	(29)	(42)								
10	0.1747	1690.22	1690.22	1690.22	1671.89	1690.22	1690.22	2	1690.22	21	1690.22	16	1650.11	3600	2.43
		(60)	(22)	(72)	(67)	(207)	(25)								

Table 9 Average performance for different phase I and phase II combinations for $n^{\text{test}} = 50$, $p = 5$, $C^{\min} = 5$ and $C^{\max} = 12$ with PR + CLS

Phase I	Phase II					
	CNS		VND		RVND	
	Avg. number of iterations	Avg. error e_j (%)	Avg. number of iterations	Avg. error e_j (%)	Avg. number of iterations	Avg. error e_j (%)
HWE	62.9	0.11	108.5	0.20	106.5	0.03
CMC	87.9	0.12	110.8	0.27	111.0	0.08

iteration at which the best solution was first found by GRASP. If $i^{\text{best}} = N^{\text{GRASP}}$, the best solution was found by PR. The column labeled t^{best} reports the amount of time spent to find the best solutions while the column t^{overall} gives the combined runtime of GRASP and PR. For problems with 5, 6 or 8 clusters, the same objective function values were found for all six options. For $p = 7$, the best solutions were found under different combinations. In 12 out of the 18 instances associated with the datasets for $p = 5, 7, 8$ clusters, PR improved the GRASP solutions by up to 1.15%; for $p = 6$, PR offered no improvement. In all cases, runtimes were well under 200 sec.

5.5 GRASP performance on the benchmark problems

In the final set of experiments, GRASP with PR was applied to a set of six benchmark instances with known optimal values. The datasets were provided by Mehrotra and Trick (1998) who solved each of them on a DEC ALPHA 3000 (Model 300) workstation with 150 MHz Alpha 21064 CPU using CPLEX 2.1 as the linear programming solver. The largest dataset contains 61 nodes and 187 edges. In their runs the number of clusters was not specified, so to duplicate that scenario, we set $p = 12$, a high enough value to ensure that we would always have a sufficient number of clusters.

The results for option (HWE, RVND) with PR and PLS activated are reported in Table 11 along with the optimal solutions for two sets of runs, the first with $C^{\min} = 0$, $C^{\max} = 450$ and the second with $C^{\min} = 0$, $C^{\max} = 512$. The number of iterations, N^{GRASP}, was set to 250 in all cases. Our best solutions ϕ^{best} are given in columns 4 and 10, the iteration number at which the best solution i^{best} was first encountered is given in columns 5 and 11, the corresponding runtimes t^{best} are given in columns 6 and 12, and the total runtimes t^{overall} are given in columns 7 and 13.

From the table, we can see that our methodology finds the exact optimum in all cases except the last in considerably less time than reported by Mehrotra and Trick. This, of course, is not surprising since we are using a much faster machine. Scaling runtimes, however, indicates that their algorithm is competitive with ours and so may be preferred for instances of the size investigated their study since it guarantees optimality. Nevertheless, it is difficult to compare performance of exact and heuristic methodologies, especially across different platforms. To a large extent the computational effort of any metaheuristic such as GRASP is proportional to the number of iterations, N^{GRASP} here, specified at the outset. A final point about the results is that PR only improved the GRASP solution for the largest instance.

Table 10 Computational results from GRASP and PR with PLS for complete USPS dataset with $n^{\text{test}} = 82$, $C^{\min} = 10$ and $C^{\max} = 20$

No. of clusters p	Phase I	Phase II	GRASP & PR + PLS				
			ϕ^{best}	PR Impr. (%)	i^{best}	t^{best} (sec)	t^{overall} (sec)
5	HWE	CNS	1577.03	0.00	116	22	99
	HWE	VND	1577.03	0.02	410*	69	127
	HWE	RVND	1577.03	0.02	264	57	135
	CMC	CNS	1577.03	0.00	41	6	93
	CMC	VND	1577.03	0.00	410*	70	96
	CMC	RVND	1577.03	0.00	18	5	113
6	HWE	CNS	1540.76	0.00	228	34	91
	HWE	VND	1540.76	0.00	185	27	79
	HWE	RVND	1540.76	0.00	228	52	140
	CMC	CNS	1540.76	0.00	44	7	91
	CMC	VND	1540.76	0.00	3	1	83
	CMC	RVND	1540.76	0.00	64	15	112
7	HWE	CNS	1371.57	1.15	410*	91	104
	HWE	VND	1367.32	1.02	410*	64	89
	HWE	RVND	1371.57	1.15	410*	89	158
	CMC	CNS	1371.57	0.99	410*	107	138
	CMC	VND	1371.57	0.31	410*	55	88
	CMC	RVND	1355.97	0.00	92	17	142
8	HWE	CNS	1148.66	0.36	410*	80	121
	HWE	VND	1148.66	0.52	410*	96	120
	HWE	RVND	1148.66	0.64	410*	85	126
	CMC	CNS	1148.66	0.00	227	34	80
	CMC	VND	1148.66	0.26	410*	62	88
	CMC	RVND	1148.66	0.13	410*	74	117

*For these instances, the best solution was found by PR in the post-processing stage after 410 GRASP iterations

6 Summary and conclusions

The reactive GRASP presented in this paper was designed to find high quality solutions to the p-capacitated clustering problem. In phase I, two efficient approaches (HWE and CMC) are presented for constructing partial initial solutions, and a dynamic restricted candidate list is proposed to then obtain feasible solutions. In the improvement phase, three neighborhoods, i.e., CNS, VND and RVND, are considered for the local search. In a post-processing step, PR is applied to overcome local optimality and to attempt to uncover even better solutions. Both CLS and PLS are implemented with PR. All components of the methodology were extensively tested on a number of instances of practical size. According to the results, HWE and CMC

Table 11 GRASP performance on benchmark problems

Graph	$\|V\|$	$\|E\|$	$C^{\min} = 0, C^{\max} = 450$						$C^{\min} = 0, C^{\max} = 512$					
			GRASP & PR + PLS				Mehrotra & trick		GRASP & PR + PLS				Mehrotra & trick	
			ϕ^{best}	i^{best}	t^{best} (sec)	t^{overall} (sec)	Solution	Time (sec)	ϕ^{best}	i^{best}	t^{best} (sec)	t^{overall} (sec)	Solution	Time (sec)
1	45	98	2928	13	1	21	2928	46.9	3238	5	<1	29	3238	79.5
2	30	56	1642	7	1	6	1642	3.0	1748	3	<1	5	1748	3.3
3	47	101	3569	27	1	23	3569	139.5	3960	6	<1	33	3960	115.8
4	47	99	1837	2	<1	32	1837	201.7	1993	4	<1	32	1993	438.4
5	30	47	1099	2	<1	9	1099	2.5	1174	1	<1	8	1174	3.0
6	61	187	22,216	250*	26	84	22,216	352.4	23,552	56	6	77	23,564	394.4

*Best solution was found by PR in the post-processing stage after 250 GRASP iterations were comparable when combined with CNS in VND, while HWE combined with RVND gave the best overall performance. However, the runtimes of the latter pair where slightly above the runtimes of the other combinations. During PR, PLS and CLS provide similar results with the latter being a bit more efficient.

In all instances tested, GRASP provided the same or better solutions than CPLEX. These results offer some assurance that GRASP can find high quality solutions when optimality cannot be established. As the number of nodes, edges, and clusters increase, the difficulty in solving model 2 optimally increases as well, often exponentially. For the 50-node, 5-cluster instances, CPLEX failed to converge within the imposed 1-hour time limit so we cannot be sure that the solutions found by our methodology are optimal. Nevertheless, that fact that we were able to optimally solve most of the benchmark problems further attests to its effectiveness.

With respect to path relinking, we can confirm the mixed results reported by others such as Boudia et al. (2006) who have performed similar analyses. For relatively small problem instances, PR offered no advantage here since GRASP was able to find the optimum without it. For larger instances, including the USPS application and some of the benchmark datasets, post-processing the GRASP solutions led to slightly better objective function values. However, the improvement was rarely significant, so it is arguable whether the procedure is justified even when using PLS instead of CLS.

References

Al-Sultan, K.S., Khan, M.M.: Computational experience on four algorithms for the hard clustering problem. Pattern Recogn. Lett. **173**, 295–308 (1996)

Ahmadi, S., Osman, I.H.: Greedy random adaptive memory programming search for the capacitated clustering problem. Eur. J. Oper. Res. **162**(1), 30–44 (2005)

Bard, J.F., Jarrah, A.I.: Large-scale constrained clustering for rationalizing pickup and delivery operations. Transp. Res. Part B: Methodol. **43**(5), 542–561 (2009)

Barreto, S., Ferreira, C., Paixao, J., Santos, B.S.: Using clustering analysis in a capacitated location-routing problem. Eur. J. Oper. Res. **179**, 968–977 (2006)

Boudia, M., Louly, M.A.O., Prins, C.: A reactive GRASP and path relinking for a combined production-distribution problem. Comput. Oper. Res. **34**(11), 3402–3419 (2006)

Brucker, J.: On the Complexity of Clustering Problem. Lecture Notes in Economics and Mathematical Systems, vol. 157, pp. 45–54. Springer, Berlin/Heidelberg (1978)

Cano, J.R., Cardon, O., Herrera, F., Sanchez, L.: A greedy randomized adaptive search procedure applied to the clustering problem as an initialization process using k-means as a local search procedure. J. Intell. Fuzzy Syst. **12**, 235–242 (2002)

Chiou, Y.-C., Lan, L.W.: Genetic clustering algorithms. Eur. J. Oper. Res. **135**(2), 413–427 (2001)

Daganzo, C.F.: Logistics System Analysis, 4th edn. Springer, Berlin (2005)

Deng, Y.: Combining mathematical programming and metaheuristics: an application to semiconductor manufacturing. Ph.D. dissertation, Graduate Program in Operations Research & Industrial Engineering, University of Texas, Austin (2009)

Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedure. J. Glob. Optim. **2**, 1–27 (1995)

Ferreira, C.E., Martin, A., de Souza, C.C., Weismantel, R., Wolsey, L.A.: The node capacitated graph partitioning problem: a computational study. Math. Program. **81**(2), 229–256 (1998)

Frank, A.: On the edge-connectivity algorithm of Nagamochi and Ibaraki. Working paper, ARTEMIS-IMAG, Université de Grenoble, Grenoble, France (1994)

Glover, F., Laguna, M., Marti, R.: Fundamentals of scatter search and path relinking. Control Cybern. **29**(3), 653–684 (2000)

Johnson, E.L., Mehrotra, A., Nemhauser, G.L.: Min-cut clustering. Math. Program. **62**(1), 133–151 (1993)

Hansen, P., Mladenovic, N.: J-means: a new local search heuristic for minimum sum of squares clustering. Pattern Recogn. **34**(2), 405–413 (2001)

Hu, B., Leitner, M., Raidl, G.R.: Combining variable neighborhood search with integer linear programming for the generalized minimum spanning tree problem. J. Heuristics **14**(5), 473–499 (2008)

Kaufman, L., Roussweuw, P.: Finding Groups in Data: An Introductory to Cluster Analysis. Wiley, New York (1990)

Karger, D.R., Stein, C.: A new approach to the minimum cut problem. J. ACM **43**(4), 601–640 (1996)

Kontoravdis, G., Bard, J.F.: A GRASP for the vehicle routing problem with time windows. ORSA J. Comput. **7**(1), 10–23 (1995)

Laporte, S., Chapleau, S., Landry, P.-E., Mercure, H.: An algorithm for the design of mailbox collection routes in urban areas. Transp. Res. Part B: Methodol. **23**(4), 271–280 (1989)

Lorena, L.A.N., Senne, E.L.F.: A column generation approach to capacitated p-median problems. Comput. Oper. Res. **31**(6), 863–876 (2004)

Mehrotra, A., Trick, M.A.: Cliques and clustering: a combinatorial approach. Oper. Res. Lett. **22**(1), 1–12 (1998)

Mladenovic, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)

Mulvey, J.M., Beck, M.P.: Solving capacitated clustering problems. Eur. J. Oper. Res. **18**(3), 339–48 (1984)

Nagmochi, H., Ibaraki, T.: Computing edge-connectivity in multigraphs and capaciated graphs. SIAM J. Discrete Math. **5**(1), 54–66 (1992)

Negreiros, M., Palhano, A.: The capacitated centered clustering problem. Comput. Oper. Res. **33**(6), 1639–1663 (2005)

Newell, G.F., Daganzo, C.F.: Design of multiple-vehicle delivery tours—I: a ring-radial network. Transp. Res. Part B: Methodol. **20B**(5), 345–363 (1986)

Osman, I.H., Ahmadi, S.: Guided construction search metaheuristics for the capacitated p-median problem with single source constraint. J. Oper. Res. Soc. **58**(1), 100–114 (2007)

Ouyang, Y.: Design of vehicle routing zones for large-scale distribution systems. Transp. Res. Part B: Methodol. **41**(10), 1079–1093 (2007)

Prais, M., Ribeiro, C.C.: Reactive GRASP: an application to a matrix decomposition problem in TDMA traffic assignment. INFORMS J. Comput. **12**(3), 164–176 (1999)

Rojanasoonthon, S., Bard, J.F.: A GRASP for parallel machine scheduling with time windows. INFORMS J. Comput. **17**(1), 32–51 (2005)

Sherali, H.D., Smith, J.C.: Improving discrete model representations via symmetry considerations. Manag. Sci. **47**(10), 1396–1407 (2001)