Computers & Operations Research 37 (2010) 1805–1813

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Near-optimal solutions for the generalized max-controlled set problem ☆

Ivairton M. Santos [a], Carlos A. Martinhon [b,*,1], Luiz S. Ochi [b]

[a] Mato Grosso Federal University, Médio Araguaia Institute, Rodovia MT-100, Km 3.5, Campus Universitário I, 78698-000 Pontal do Araguaia, MT, Brazil
[b] Fluminense Federal University, Department of Computer Science, Rua Passo da Pátria, 156 - Bloco E - 3 andar - Boa Viagem, 24210-240 Niterói, RJ, Brazil

ARTICLE INFO

Available online 2 February 2010

Keywords:
Sandwich graph problems
Approximation algorithms
Linear programming
Heuristic
Tabu search

ABSTRACT

In this work we deal with sandwich graphs $G=(V,E)$ and present the notion of vertices f-controlled by a subset $M \subseteq V$. We introduce the generalized max-controlled set problem (GMCSP), where minimum gaps (defined by function f) and positive weights are associated to each vertex of V. In this case, the objective is to find a sandwich graph G in order to maximize the sum of the weights associated to all vertices f-controlled by M. We present a $\frac{1}{2}$—approximation algorithm for the GMCSP and a new procedure for finding feasible solutions based on a linear relaxation. These solutions are then used as starting point in a local search procedure (Tabu Search with Path Relinking) looking for solutions of better quality. Finally, we present some computational results and compare our heuristics with the optimum solution value obtained for some instances of the problem.

1. Introduction

Given two graphs $G_1=(V,E_1)$ and $G_2=(V,E_2)$ such that $E_1 \subseteq E_2$, we say that $G=(V,E)$, where $E_1 \subseteq E \subseteq E_2$, is a *sandwich graph* for some property Π if $G=(V,E)$ satisfies Π. A *sandwich problem* consists of deciding whether there exists some sandwich graph satisfying Π. We call *fixed edges* and *optional edges*, respectively, the edges belonging to E_1 and $E_2 \backslash E_1$. Many different properties may be considered in the context of sandwich graphs as, for example, physical mapping of DNA, temporal reasoning, synchronizing parallel processes, phytogenetic trees, sparse systems of linear equations, among others [4,11,12].

For instance, in the chordal sandwich problem we require G to be a chordal graph (a graph where every cycle of length at least four possesses a chord—an edge linking two non-consecutive vertices in the cycle). The chordal sandwich problem is closely related to the minimum fill-in problem [24]: given a graph G, the objective is to find the minimum number of edges to be added to G so that the resulting graph is chordal. The minimum fill-in problem has applications to areas such as solution of sparse systems of linear equations [21]. Another important sandwich problem is the interval sandwich problem, where we require the sandwich graph G to be an interval graph (a graph whose vertices are in a one-to-one correspondence with intervals on the real line in such a way that there exists an edge between two vertices if and only if the corresponding intervals intersect). Kaplan and Shamir [16] describe applications to DNA physical mapping via the interval sandwich problem.

Given an undirected graph $G=(V,E)$ and a set of vertices $M \subseteq V$, a vertex $v_i \in V$ is said to be *controlled* by M if $|N_G[v_i] \cap M| \geq |N_G[v_i]|/2$, where $N_G[v_i]=\{v_i\} \cup \{v_j \in V | [v_i,v_j] \in E\}$. The set M defines a *monopoly* in G if every vertex $v_i \in V$ is controlled by M. Therefore, subset M (also referred as *coalition*) defines a monopoly in G, if and only if, *cont*$(G,M)=V$, where *cont*(G,M) denotes the set of vertices controlled by M in G. As discussed in [3,5,14,19,20], the notion of monopoly has applications to local majority voting in distributed environments, agreement in agent systems, among others. In [3,20], many different questions regarding monopolies or coalitions in graphs are considered, as for example: (a) What is the maximum influence of M (as a function of $|M|$)? (b) How small can a monopoly be? In all these papers, different definitions for "neighborhood" and "majority" can be used, without affecting the results.

Prior to define the generalized max-controlled set problem (GMCSP), we first consider the monopoly verification problem (MVP) and the max-controlled set problem (MCSP), as introduced in [17]. Basically, in all these problems the coalition M remains unchanged and the idea is to find a (smart) way of "controlling" a given set of objects by modifying the system topology. Thus, in the MVP, given a set $M \subseteq V$ and two graphs $G_1=(V,E_1)$ and $G_2=(V,E_2)$, where $E_1 \subseteq E_2$, the question is to decide whether there exists a sandwich graph $G=(V,E)$ such that $E_1 \subseteq E \subseteq E_2$ and M is a

☆ A preliminary version of this paper appeared in the Proceedings of the IV Latin-American Algorithms, Graphs and Optimization Symposium, IV LAGOS, Puerto Varas, Chile, Electronic Notes in Discrete Mathematics, vol. 30, pp. 183–188, 2008.

* Corresponding author. Tel.: +55 21 2629 5644; fax: +55 21 2629 5669.
E-mail addresses: ivairton@ufmt.br (I.M. Santos), mart@dcc.ic.uff.br (C.A. Martinhon), satoru@ic.uff.br (L.S. Ochi).

[1] Sponsored by CNPq/Brazil and FAPERJ/Brazil.

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.01.012

monopoly in G. If the answer of the MVP is "yes", the authors in [17] show how to find a minimum (resp., maximum) number of optional edges such that M remains a monopoly of G within the new topology defined by E. They proved that both problems, (resp., named min-neighborhood and max-neighborhood monopolies) can be solved in polynomial time in the size of E_2.

If the answer of the MVP above is "no", the authors in [17] propose the MCSP, whose goal is to find a set E such that $E_1 \subseteq E \subseteq E_2$ and the number of vertices controlled by M in G=(V,E) is maximized. The MVP can be solved in polynomial time by formulating it as a network flow problem. Unfortunately, the MCSP is NP-hard even for those instances where G_1 is an empty graph and G_2 is a complete graph (see [17]). Finally, through randomized rounding and derandomization techniques, Martinhon and Protti [18] proposed, for $n>4$, a non-trivial $(1/2)+(1+\sqrt{n})/2(n-1)-$approximation algorithm for the MCSP (improving the previous $\frac{1}{2}-$approximation algorithm proposed in [17]).

Now we present the definition of vertices f-controlled by a subset $M \subseteq V$ (as introduced by Makino et al. [17]). Given a value $f_i \in \mathbb{Z}$, the vertex $v_i \in V$ is said to be *f-controlled* by M, if and only if, $|N_G[v_i] \cap M| - |N_G[v_i] \cap U| \geq f_i$, where $U = V \backslash M$. The constant f_i represents the minimum gap necessary to f-control vertex $v_i \in V$. Note, for instance, that for some constant $A > |V|$ with $f_i = -A$ (resp., $f_i = +A$) the vertex v_i is always (resp., never) f-controlled by M at every sandwich graph G. We denote by *minimum gap* of a vertex v_i, the value f_i associated to $v_i \in V$.

Therefore, in the GMCSP, we are given a subset $M \subseteq V$, two graphs G_1=(V,E_1), G_2=(V,E_2) with $E_1 \subseteq E_2$, weights $w_i \geq 0$ and minimum gaps $f_i \in \mathbb{Z}$ for each vertex $v_i \in V$. Our goal in the GMCSP is to find a sandwich graph G in order to maximize the sum of the weights associated to all vertices f-controlled by M. The GMCSP is obviously NP-hard since it generalizes the MCSP (particular instance of the GMCSP for f_i=0 and w_i=1, $\forall v_i \in V$). To better illustrate the GMCSP, consider, for example, an application where *two* different groups of individuals are associated to companies, polling system or political parties. In real life situations, the associated weights and minimum gaps may indicate, respectively, the importance and degree of agreement of an individual or agent related to a given proposal. In these cases, we hope to establish or destroy some partnerships (represented by the set of optional edges) in order to maximize the importance of all individuals in accordance with the proposal. In our model, stronger and durable partnerships may be represented by the set of fixed edges.

Our paper is organized as follows. In the Section 2, we present some reduction rules in order to reduce the size of the GMCSP. In the Section 3, we present a $\frac{1}{2}-$approximation algorithm (named, *BasedMYK*) and a based linear programming heuristic (named, *BasedLP*) for the GMCSP. In Section 4, the best solution obtained by both *BasedMYK* and *BasedLP* procedures is used as starting point in our Tabu Search with Path Relinking procedure. Finally, we present our computational results at Section 5 and some final comments at Section 6.

2. Reduction rules

Now we generalize the reduction rules applied to the MCSP (as described in [17,18]). In our case however, we change the definition of controlled by f-controlled vertices. As it will be observed later, besides reducing the total number of optional edges, these rules will also help us to define a tight linear integer programming formulation for the GMCSP.

Let G_1=(V,E_1) and G_2=(V,E_2) be two graphs with $E_1 \subseteq E_2$ as above. For $A,B \subseteq V$, we denote by $D(A,B) = \{[v_i,v_j] \in E_2 \backslash E_1 \mid v_i \in A, v_j \in B\}$, the set of optional edges with endpoints belonging to A and B, respectively. Given $M \subseteq V$, two rules are used: a new edge set E_1^* is obtained by the union of E_1 and D(M,M) and a new edge set E_2^* is obtained by removing D(U,U) from E_2, where $U = V \backslash M$. Therefore, the set E in the sandwich graph G must satisfy: $E_1 \cup D(M,M) \subseteq E \subseteq E_1 \cup D(M,M) \cup D(U,M)$. For simplicity, assume from now on E_1=E_1^* (Reduction Rule 1) and E_2=E_2^* (Reduction Rule 2).

Prior to describe the remaining reduction rules, consider the following partition of V: we denote by M_{AC} and U_{AC}, respectively, the subset of vertices belonging to M and U which are always f-controlled by M in any sandwich graph G. Analogously, we denote by M_{NC} and U_{NC} the subset of vertices which are never f-controlled by M in any sandwich graph. Finally, we define the subsets $M_R = M \backslash (M_{AC} \cup M_{NC})$ and $U_R = U \backslash (U_{AC} \cup U_{NC})$. Clearly, $M_R = \emptyset$ if and only if $U_R = \emptyset$. Then we add the assumption $M_R, U_R \neq \emptyset$.

In order to construct the partition of V as above, it is sufficient to look at all "worst case" assignments. Thus, after setting E=E_2 we identify all vertices belonging to $M_{AC} \cup U_{NC}$. Similarly, if we set E=E_1, we determine $U_{AC} \cup M_{NC}$. The reduction rules are listed in the sequel.

- Add to E_1 all edges belonging to $D(M_{AC} \cup M_{NC}, U_R)$ (*Reduction Rule* 3).
- Remove from E_2 all edges belonging to $D(M_R, U_{AC} \cup U_{NC})$ (*Reduction Rule* 4).
- Add or remove at random the edges belonging to $D(M_{AC} \cup M_{NC}, U_{AC} \cup U_{NC})$ (*Reduction Rule* 5).

The reduction rules are summarized in Fig. 1. Notice that, since the status of subsets M_{AC} and M_{NC} in the third reduction rule (respectively, U_{AC} and U_{NC} in the fourth reduction rule) remains unchanged, we try to increase the chances of f-controlling vertices of $U_R \subseteq U$ (respectively, $M_R \subseteq M$).

Moreover, observe that reduction rules 3 and 4 can be alternately applied until no improvements are possible, as illustrated in the example of Fig. 2. In this example, we consider f_i=0, w_i=1, $\forall v_i \in V$ and M={a,b,c,d}. Note that graph of Fig. 2(b) is obtained after execution of the third rule over the graph of Fig. 2(a), and graph of Fig. 2(c) is obtained after execution of the fourth rule over the graph of Fig. 2(b). At this point, instead of applying rule 5, we alternately execute rules 3 and 4 until no improvements are possible. The resulting graph is depicted in Fig. 2(d) (without optional edges).

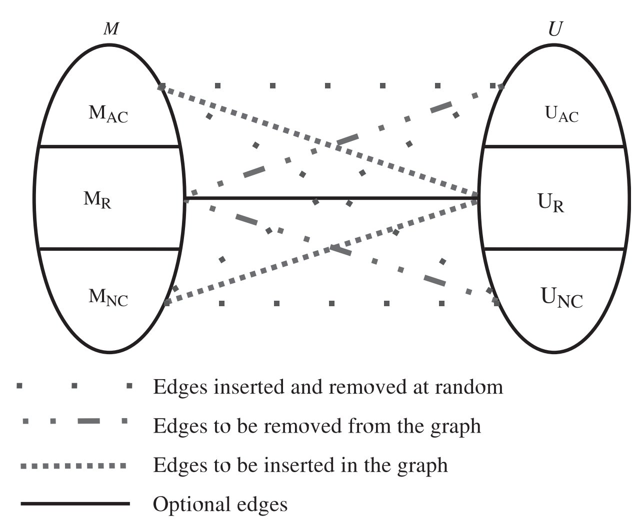

Fig. 1. Reduction rules 3–5.

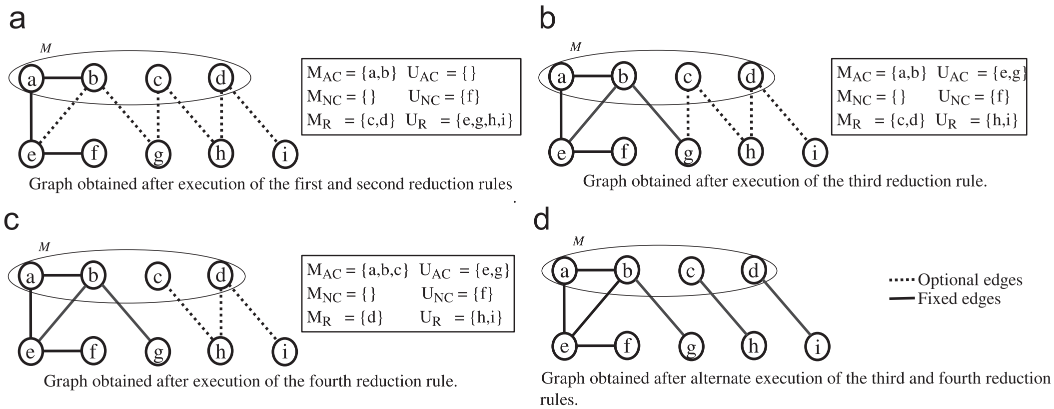

Fig. 2. Alternate execution of rules 3 and 4.

In the next section we present an approximation algorithm and show how to use the reduction rules in order to obtain a tight linear integer programming formulation for the GMCSP.

3. Generalized max-controlled set problem—GMCSP

3.1. *A* $\frac{1}{2}$–approximation *algorithm for the GMCSP*

The $\frac{1}{2}$–approximation algorithm for the MCSP proposed by Makino et al. [17] may be easily extended to the GMCSP and is described in the sequel. First of all, consider W_1 and W_2 the sum of the weights associated to all vertices *f*-controlled by *M* in *G*=(*V*,*E*) for E=E_1 and E=E_2 respectively. The variable $z_{H\ 1}$ denotes the value of the best solution obtained in both cases. Thus, we have the following approximation algorithm for the GMCSP (denoted by *BasedMYK*, for short):

Algorithm 1. BasedMYK—$\frac{1}{2}$–approximation algorithm for the GMCSP.

1: $W_1 \leftarrow$ Sum of the weights associated to all vertices *f*-controlled by *M*, obtained by removing $D(U,M)$ from E_2;
2: $W_2 \leftarrow$ Sum of the weights associated to all vertices *f*-controlled by *M*, obtained by adding $D(U,M)$ to E_1;
3: $z_{H1} \leftarrow \max\{W_1, W_2\}$;

Theorem 1. *Let* z_{max} *be the optimum solution value of the* GMCSP. *Then* $z_{H1} \geq \frac{1}{2} z_{max}$.

Proof. Since all weights are positive, one directly observes that $z_{max} \leq W_1 + W_2 \leq 2 \times \max\{W_1, W_2\}$. Now, since $z_{H\ 1}$=max{W_1, W_2} it follows that $z_{H1} \geq \frac{1}{2} z_{max}$. □

3.2. *An heuristic based on a linear relaxation for the GMCSP*

In order to describe our based linear programming heuristic for the GMCSP (denoted by *BasedLP*, for short), we introduce the following integer programming formulation. We define binary variables $z_i \in \{0,1\}$ for every $v_i \in V$, which determine whether or not vertex v_i is *f*-controlled by *M*. Binary variables x_{ij} are used to decide whether optional edges [v_i,v_j] belonging to $E_2 \backslash E_1$ will be included or not in the sandwich graph. The constants $w_i \in \mathbb{Z}^+$ and $f_i \in \mathbb{Z}$ denote, respectively, the positive weight and minimum gap of vertex $v_i \in V$. The constants $a_{ij} \in \{0,1\}$ are associated to $[v_i, v_j] \in E_2$ with a_{ij}=1, if and only if, *i*=*j* or $[v_i, v_j] \in E_2$. Further, we assume that a_{ij}=a_{ji}, $\forall v_i, v_j \in V$.

Now, consider $K = |U| + \max\{|f_i|, \forall v_i \in V\}$ for $U = V \backslash M$. Then, we have the following linear programming formulation for the GMCSP:

$$z_{max} = \text{Maximize} \sum_{v_i \in V} w_i z_i \tag{1}$$

subject to:

$$\frac{\sum_{v_j \in M} a_{ij} x_{ij} - \sum_{v_j \in U} a_{ij} x_{ij} - f_i}{K} + 1 \geq z_i, \quad \forall v_i \in V \tag{2}$$

$$x_{ij} = 1, \quad \forall [v_i, v_j] \in E_1 \tag{3}$$

$$x_{ii} = 1, \quad \forall v_i \in V \tag{4}$$

$$x_{ij} \in \{0,1\}, \quad \forall [v_i, v_j] \in E_2 \backslash E_1 \tag{5}$$

$$z_i \in \{0,1\}, \quad \forall v_i \in V \tag{6}$$

In the formulation above the objective function (1) computes the sum of the weights of all vertices *f*-controlled by *M*. The inequality (2) guarantees the *f*-control of vertex v_i whenever the left side of (2) is greater or equal than 1 (otherwise, vertex v_i is not *f*-controlled by *M* and z_i can be settled to 0). The division by *K* maintains the quotient in the left side of (2) always greater or equal than -1, ensuring $z_i \geq 0$. The equalities (3) and (4) define the set of fixed edges in G_1. A linear relaxation, represented by $\tilde{P}$, can be obtained by substituting constraints (5) and (6) (associated to binary variables) by $x_{ij} \in [0,1]$ and $z_i \in [0,1]$, respectively.

Now, consider the sets M_R, M_{AC}, M_{NC}, U_R, U_{AC}, U_{NC} as described in the reduction rules (see Section 2). In order to construct a tight formulation for the GMCSP, we define a constant b_i for each vertex $v_i \in M_R \cup U_R$ through the following auxiliary equality:

$$b_i = \left| \sum_{v_j \in M} a_{ij} x_{ij} - \sum_{v_j \in U} a_{ij} x_{ij} - f_i \right| \quad \text{for } v_i \in M_R \cup U_R \tag{7}$$

The constant b_i is computed in the following way: if $v_i \in M_R$, we set $x_{ij} = 1, \forall [v_i, v_j] \in E_2 \backslash E_1$. Analogously, if $v_i \in U_R$, we set $x_{ij} = 0, \forall [v_i, v_j] \in E_2 \backslash E_1$. Obviously, in both cases we must have $x_{ij} = 1, \forall [v_i, v_j] \in E_1$.

Thus, a stronger formulation for the GMCSP can be obtained by changing constraint (2) by constraints (8)–(10) as described in the sequel:

$$\frac{\sum_{v_j \in M} a_{ij} x_{ij} - \sum_{v_j \in U} a_{ij} x_{ij} - f_i}{b_i} + 1 \geq z_i, \quad \forall v_i \in M_R \cup U_R \tag{8}$$

$$z_i = 1, \quad \forall v_i \in M_{AC} \cup U_{AC} \tag{9}$$

$$z_i = 0, \quad \forall v_i \in M_{NC} \cup U_{NC} \tag{10}$$

Let $\overline{P}$ be the linear relaxation associated to this new formulation. Similarly to constraint (2), constraint (8) above also guarantees the f-control of vertex v_i by M whenever the first part of inequality (8) is greater than or equal to 1. Moreover, the division by b_i always guarantees $z_i \geq 0$. The equality (9) exhibits all vertices always f-controlled by M and equality (10) the vertices never f-controlled by M. Formally, we can prove the following result:

Theorem 2. *Let $\tilde{z}_{max}$ and $\overline{z}_{max}$ be the optimum values of $\tilde{P}$ and $\overline{P}$, respectively. Then, $\tilde{z}_{max} \geq \overline{z}_{max}$.*

Proof. Firstly, note by definition of K and b_i (see equality (7)) that we have $b_i \leq K, \forall v_i \in V$. Now, let $(\tilde{x},\tilde{z})$ and $(\overline{x},\overline{z})$ be two optimum fractional solutions of both $\tilde{P}$ and $\overline{P}$, respectively. If vertex v_i is f-controlled by M at some integer feasible solution than both $\tilde{z}_i = \overline{z}_i = 1$ (see inequalities (2) and (8)). However, if v_i is a vertex not f-controlled by M, it follows by definition of K and b_i that $\tilde{z}_i \geq \overline{z}_i$, $\forall v_i \in V$. Therefore $\sum_{v_i \in V} w_i\tilde{z}_i \geq \sum_{v_i \in V} w_i\overline{z}_i$. □

It is assumed from now on that $(\overline{x},\overline{z})$ will denote an optimal solution of the strengthened formulation $\overline{P}$ and $\overline{z}_{max}$ its associated objective function value. The optimum solution value of the GMCSP is denoted by z_{max}.

A feasible solution for the GMCSP can be constructed through the linear relaxation in the following way. Given $(\overline{x},\overline{z})$ of $\overline{P}$ with components $\overline{x}_{ij} \in [0,1]$, $\forall [v_i,v_j] \in E_2$ and $\overline{z}_i \in [0,1]$, $\forall v_i \in V$, we define as f-controlled all vertices $v_i \in V$ with $\overline{z}_i = 1$, and as not f-controlled (or uncontrolled) the remaining vertices of V with $\overline{z}_i < 1$. However, how to decide about the set of optional edges with $\overline{x}_{ij} \in (0,1)$? Actually, the following result ensure that binary values $\overline{x}_{ij} \in \{0,1\}$ are always obtained after solving the linear relaxation $\overline{P}$. As a consequence of that, after determining which vertices are f-controlled or not by M, we have a straightforward procedure to decide which of the optional edges will be selected or not in our *BasedLP* heuristic. Formally, we have established the following result (proved in the Appendix):

Theorem 3. *Consider $(\overline{x},\overline{z})$ an optimum solution of $\overline{P}$ with components $0 \leq \overline{x}_{ij} \leq 1$, $\forall [v_i,v_j] \in E_2$, $\overline{z}_i \in [0,1]$; $\forall v_i \in V$ and optimum value $\overline{z}_{max} = \sum_{i \in V} w_i\overline{z}_i$. If $0 < \overline{x}_{ij} < 1$ for some edge $[v_i,v_j] \in E_2 \backslash E_1$, we have another feasible solution $(\tilde{x},\tilde{z})$ of $\overline{P}$ with $\overline{z}_{max} = \sum_{v_i \in V} w_i\tilde{z}_i$ where $\tilde{x}_{ij} \in \{0,1\}, \forall [v_i,v_j] \in E_2$ and $\tilde{z}_i \in [0,1], \forall v_i \in V$.*

Theorem 3 above is illustrated in the example of Fig. 3. Consider w_i=w_j and f_i=f_j=0, $\forall v_i, v_j \in V$. In both cases, we have the objective function values equal to w_0+w_1+w_3+w_4 (note that vertex 2 is never f-controlled by M).

Within the proof of Theorem 3 above, we can show that fractional values defining probabilities are only possible for instances with $w_v/b_v = w_u/b_u$ and $[u,v] \in E_2 \backslash E_1$ (see the Appendix for the proof). Note in the example of Fig. 3 that these conditions are verified since w_i=w_j and b_i=b_j=1, for every v_i=v_j of V.

A new algorithm named *Best*(*MYK+LP*), may be constructed by just choosing the best solution obtained in the *BasedMYK* and *BasedLP* procedures. As discussed in [18] for the MCSP (particular instance of the GMCSP for f_i=0 and w_i=1, $\forall v_i \in V$), the *Best*(-*MYK+LP*) procedure has a performance ratio equal to $(1/2)+(1+\sqrt{n}/2(n-1))$, $\forall n > 4$. In [18], the authors propose a randomized rounding and a derandomization procedure for the MCSP using probabilities defined by the linear relaxation of $\tilde{P}$ (in their formulation w_i=1, f_i=0 and b_i=K, $\forall v_i \in V$). Thus, as a consequence of Theorem 3 (when restricted to MCSP), the derandomization step proposed in [18] is not necessary, since after solving $\tilde{P}$ by the simplex algorithm, we always obtain 0–1 assignments for all coordinates of vector $\tilde{x}$.

4. A Tabu Search with Path Relinking for the GMCSP

Given a function $h(\cdot)$ to be maximized over a set Ψ of feasible solutions, *Tabu Search-TS* procedure starts from an initial point in Ψ and proceeds iteratively from one point in Ψ to another until some stop condition is met. To each solution $S \in \Psi$, one associates a set of neighbors $N(S) \subset \Psi$. Basically, TS combines local search in this neighborhood with a number of clever anti-cycling rules which prevent the search from getting "confined" in local optima [7]. Tabu Search procedure also includes *intensification* and *diversification* mechanisms by forcing the search, resp., into promising regions or previously unexplored areas of the search space. It is usually based on some form of *short-term* or *long-term memory* of the search (such as a frequency memory). For further details about TS and different intensification and diversification strategies see [6,13,9].

In the *Path Relinking* (*PR*), the basic idea is to generate solutions of better quality by exploring trajectories connecting solutions of an *elite set* $\Im$ of selected solutions. The PR was first introduced in [7] and can be viewed as a strategy that seeks to incorporate attributes of $\Im$ by favoring these attributes in the selected moves. A more thorough description of PR can be found in [8,10]. Some applications combining TS with PR can be found in [15,2], resp., for the Vehicle Routing and the Steiner problem in graphs.

Given a sandwich graph G=(V,E) (associated to a current solution S), we denote by $N_0(G)$, our neighborhood structure to be used within the TS framework [6,7,9]. In this neighborhood, we hope to f-control new vertices with positive weights in such a way that all vertices f-controlled by M in G remains f-controlled after the local search.

Assume, without loss of generality that $V = M_R \cup U_R$. Thus, given a sandwich graph G, consider $M_G \subseteq M_R$ and $U_G \subseteq U_R$ the subset of vertices f-controlled by M in G. Also consider the following auxiliary notation: $L_G = M_G \cup U_G$ and $L_D = (M_R \cup U_R) \backslash L_G$. In addition, given a sandwich graph G, we call *current gap* of a vertex $v_i \in V$ the value $f_G(v_i) = |N_G[v_i] \cap M| - |N_G[v_i] \cap U| - f_i$. Note for instance that vertex $v_i \in V$ is f-controlled by M in G, if and only if, $f_G(v_i) \geq 0$. Therefore, $f_G(v_i) < 0, \forall v_i \in L_D$. Finally, given a vertex $v_i \in L_D$, we denote by $H(v_i) = \{v_j \in V | [v_i,v_j] \in E_2 \backslash E_1$ and $f_G(v_j) \neq 0\}$, the set of v_i neighbors that may help in its f-controlling.

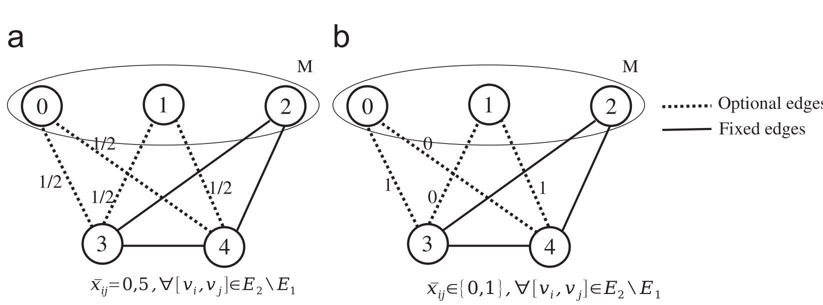

Fig. 3. Two optimal solutions with fractional and integer coordinates, respectively.

As discussed before, given a sandwich graph G, our goal in the neighborhood $N_0(G)$ is to f-control new vertices in L_D maintaining all vertices in L_G (set of vertices already f-controlled by M) as controlled. Our local search procedure for the GMCSP is then summarized in Algorithm 2.

Algorithm 2. Local search for the GMCSP.

1: Given a sandwich graph G=(V,E) as input data
2: **while** $L_D \neq \emptyset$ **do**
3: Pick at random $v_i \in L_D$
4: **if** $v_i \notin T$ **then**
5: **if** $|H(v_i)| \geq |f_G(v_i)|$ **then**
6: **while** $f_G(v_i) < 0$ **do**
7: $v_j \leftarrow$ Randomly choose a vertex in $H(v_i)$
8: **if** $v_i \in M_R$ **then**
9: $E \leftarrow E \backslash \{[v_i, v_j]\}$
10: **else**
11: $E \leftarrow E \cup \{[v_i, v_j]\}$
12: **end if**
13: $f_G(v_i) \leftarrow f_G(v_i)+1$
14: $f_G(v_j) \leftarrow f_G(v_j)-1$
15: $H(v_i) \leftarrow H(v_i) \backslash \{v_j\}$
16: **end while**
17: **end if**
18: **end if**
19: $L_D \leftarrow L_D \backslash \{v_i\}$
20: **end while**
21: *return G*

In the line 1, graph G may be generated by our *Best*(*MKY+LP*) procedure (see Section 3.2) or during the execution of the TS procedure. In lines 2–4 we choose vertices v_i from L_D whenever this vertex is not present in the solutions implicitly defined by the tabu list (represented by T). Note that vertex v_i can be f-controlled by M in $N_0(G)$, if $|f_G(v_i)|$ is less or equal than $|H(v_i)|$ (line 5). Therefore, it suffices to add/remove a convenient number of optional edges (equal to $|f_G(v_i)|$) in order to guarantee the f-control of $v_i \in L_D$. In addition, from lines 6–16, note that if a vertex v_i to be f-controlled by M belongs to $L_D \cap M_R$ (respectively $L_D \cap U_R$) it will be necessary to remove (respectively, to add) $|f_G(v_i)|$ optional edges incident to v_i. Further, we update the current gap of vertex v_i and its associated neighbors $v_j \in H(v_i)$ (lines 13 and 14), and update sets $H(v_i)$ and L_D (lines 15 and 19). Note, for instance, that all vertices $v_j \in H(v_i)$ remains f-controlled or non-f-controlled by M after each updating.

The tabu list T (subset of L_D) is constructed in the following way: given a sandwich graph G representing a local optimal, we choose an arbitrary vertex v_i (f-controlled by M) to be removed from L_G. If $v_i \in M_G$ (respectively $v_i \in U_G$) we define a new solution by adding (by removing) all edges incident to vertex v_i and updating all associated costs. This vertex remains in the tabu list by $|T|$ steps. The size of the tabu list will be discussed later in our computational results.

Other very natural neighborhood structures may be considered for the GMCSP (represented here by $N_\ell(G)$ for $\ell \geq 1$). Given a sandwich graph G, the local search may be performed, for example, by removing ℓ vertices f-controlled by M in G and f-controlling new vertices whenever some improvement in the objective function value is attained. Computational tests, however, indicated that these neighborhood structures were not efficient when compared with $N_0(G)$ [23].

We combine TS and PR by trying to incorporate good attributes of our best solutions (elite set $\Im$). The main steps of our TS with PR are summarized in Algorithm 3. In the line 1, the initial solution G is generated by our *Best*(*MKY+LP*) (as in the Section 3.2). In the line 2 we initialize, resp., the best solution G^*, the value of the best solution G'' obtained through Aspiration Criteria and the elite set $\Im$. Parameters i_{max} and j_{max} denote, resp., the total number of steps in the Tabu Search and Local Search procedures. Between lines 7–11, we compute the best solution obtained by both Local Search and Aspiration Criteria (graph G'). Note in the Aspiration Criteria that movements using vertices of the tabu list are allowed whenever some improvement in the objective function is possible (line 8). Between lines 12–15 we restart our Local Search whenever some best solution G^* is found. In the lines 16–17 we update the elite set $\Im$, and in the lines 19–20 we update the set of f-controlled vertices $L_{G'}$ and the tabu list T. Finally, we have the Diversification and Path Relinking procedures in the lines 23–24.

Algorithm 3. Tabu search with Path Relinking for the GMCSP.

1: Given an initial sandwich graph G=(V,E) as input data
2: $G^\star \leftarrow G$, $z_H(G'') \leftarrow -\infty$, Initialize elite set $\Im \leftarrow \emptyset$;
3: $i \leftarrow 0, j \leftarrow 0$ {Auxiliary variables}
4: **while** $i < i_{max}$ **do**
5: $T \leftarrow \emptyset$ {Initialize the tabu list}
6: **while** $j < j_{max}$ **do**
7: $G' \leftarrow LocalSearch(N_0(G) \backslash T)$
8: $G'' \leftarrow LocalSearch(N_0(G) \cap T)$ through Aspiration Criteria
9: **if** $z_H(G'') > z_H(G')$ **then**
10: $G' \leftarrow G''$
11: **end if**
12: **if** $z_H(G') > z_H(G^\star)$ **then**
13: $G^\star \leftarrow G'$
14: $j \leftarrow 0$
15: **end if**
16: **if** $z_H(G') >$ "worst solution in the elite set $\Im$" **then**
17: Add G' to the elite set $\Im$
18: **end if**
19: Choose, at random, a subset $P \subseteq L_{G'}$
20: Update subsets $L_{G'} \leftarrow L_{G'} \backslash P$ and $T \leftarrow T \cup P$
21: $j \leftarrow j+1$
22: **end while**
23: $G^\star \leftarrow$ Path Relinking($\Im$)
24: $G \leftarrow Diversification(G')$
25: $i \leftarrow i+1$
26: **end while**
27: *Return* $G^\star$

Notice that our intensification strategy operates by re-starting the local search from high quality solutions (line 14). In the line 19, the size of the P (subset of $L_{G'} = M_{G'} \cup U_{G'}$) can be empirically defined. In our tests, we have used $|P| = 1$, but other sets with $|P| \geq 2$ were also considered without a suggestive improvement. If $|P| \geq 2$, a good choice is to choose $P \subseteq M_{G'}$ or $P \subseteq U_{G'}$, otherwise, if P contains vertices of both $M_{G'}$ and $U_{G'}$, the simultaneous removal of all vertices of P could be more complicated and computationally expensive. Thus, to change the status of a single vertex $v \in M_G$ (resp., $v \in U_G$) from f-controlled to non-f-controlled, we just add (resp., remove) all optional edges incident to v. All parameters involved, as the size of $\Im$ and T, and the number of iterations i_{max} and j_{max} were empirically defined. As a final remark, note that only TS search may be accomplished (without PR) by just removing lines 16, 17 and 23 of Algorithm 3.

In the next, we describe our Diversification and Path Relinking procedures.

4.1. Diversification phase

After the intensification step, we hope to diversify the search looking for unexplored regions of the feasible set [6,7]. In our case, when the diversification takes place the current solution is moved to a randomly selected part of the search space which has not been thoroughly searched.

In our implementation of the GMCSP, this "degree" of diversification is related to the number of vertices involved in the process. Thus, given a sandwich graph G and a parameter k, our diversification is performed by randomly choosing a subset $K \subseteq L_G$ (where $|K| \leq k$) of vertices f-controlled by M in G.

An arbitrary order of these $|K|$ vertices is chosen and they are settled as non-f-controlled one by one following this order. Hence, if $v \in M_G \cap K$ (respectively, $v \in U_G \cap K$) we can add (respectively, remove) all optional edges incident to v. Obviously, we must update the current gaps of the associated neighbors of v. After that, this solution (seed) is then used as starting solution for a new local search procedure as described in the preceding section.

4.2. Path Relinking for the GMCSP

As mentioned earlier, PR explore trajectories between elite solutions (represented here by $\Im$). Within the PR framework (see [9,10]), we hope to improve the best solutions gathered by our TS procedure. In our case, PR may be viewed as post-optimization procedure and is applied whenever the intensification phase is concluded [22].

Given a sandwich graph G, we associate an attribute vector A with components 1 and 0 denoting, respectively, the vertices f-controlled and non-f-controlled by M in G. Thus, in order to define a trajectory between two solutions of $\Im$, we consider two sandwich graphs G_{source} and G_{dest} (resp., *source* and *destination*) belonging to $\Im$ with the biggest diversity between them (i.e., number of distinct components of A).

Therefore, if A_{source} and A_{dest} are two associated vectors of binary 0–1 values, we hope to find better solutions by following a trajectory from A_{source} and A_{dest} (equivalently between G_{source} and G_{dest}) by incorporating, whenever it is possible, the good attributes from both targets. The main steps of our PR are summarized in Algorithm 4.

Algorithm 4. Path Relinking for the GMCSP.

1: From G_{source} and G_{dest}, construct attributes A_{source} and A_{dest}
2: $R \leftarrow$ set of the different attributes between A_{source} and A_{dest}
3: **while** $R \neq \emptyset$ **do**
4:   $A', A'' \leftarrow A_{source}$
5:   **for** every $i \in R$ **do**
6:     $A'[i] \leftarrow A_{dest}[i]$
7:     **if** $cost(A') \geq cost(A'')$ **then**
8:       $A'' \leftarrow A'$ and $j \leftarrow i$
9:     **end if**
10:     $A'[i] \leftarrow A_{source}[i]$
11:   **end for**
12:   $A_{source} \leftarrow A''$
13:   **if** $cost(A_{source}) > cost(A^{\star})$ **then**
14:     $A^{\star} \leftarrow A_{source}$
15:   **end if**
16:   $R \leftarrow R \backslash \{j\}$
17: **end while**
18: Return $G^{\star}$

In the line 1, we construct attribute vectors A_{source} and A_{dest} associated respectively to G_{source} and G_{dest}. In the line 2, we build the subset R of indices denoting the different attributes from both targets. In the line 4, we initialize auxiliary vectors A' and A'' to be used in the construction of our trajectory. Between lines 5–11, we compute a new solution A'' by incorporating it the best attribute present in A_{dest}. This may be accomplished in the following way. Firstly, in the line 6, we create a new temporary solution A' with attribute $A'[i] = A_{dest}[i]$. To do that and change an attribute of a vertex $v \in M_G$ (resp., $v \in U_G$) from f-controlled to non-f-controlled (resp., to non-f-controlled to f-controlled) we use the same set of edges incident to v and present in the target destination G_{dest}. The remaining attributes of A' are updated accordingly. Every time an improvement is attained (i.e., we maximize the objective function value), we save this new solution in A'' and its corresponding attribute j (lines 7 and 8). The solution A' is restored in the line 10 and the process is repeated. In the line 12 we update A_{source} and in the lines 13, 14 we update the best attribute vector $A^{\star}$. Finally, in the line 16 we remove index j and repeat the overall process until $R = \emptyset$ (lines 3–17). We conclude in the line 18 by returning the best solution $G^{\star}$ associated to $A^{\star}$.

In our tests we have settled $G_{dest} = G^{\star}$, $G_{source} = G'$ (arbitrarily chosen at $\Im$) and $|\Im| = 10$. Different combinations of our targets and $|\Im|$ were also considered without an indicative improvement.

5. Some computational results

In order to evaluate our construction heuristics and local search procedure for the GMCSP, we choose the *open source* package GNU/GLPK version 4.8, for generating exact solutions of instances with 50, 75 and 100 vertices, respectively. All proposed algorithms were implemented in the Language C with *gcc* compiler version 3.3.2 in the Linux platform (Mandrake 9.1 and Fedora Core 2 distribution). The tests were performed in similar platforms with processor Pentium IV, 2.60 GHz with 512 Mb of RAM memory in a shared computer.

All instances considered here were generated at random for parameters empirically defined and tested. The probability of choosing a vertex of M was settled around 27%. An edge belongs to E_2 with probability 80%, and, between them, 70% were selected as optional edges. All vertices have associated weights and minimal gaps defined at random within intervals established at hand. The selected parameters are represented in the label of each instance according to the following example: if G100-20-10-01 is the label of an instance, we have 100 vertices, with intervals of weights varying between 1–20 and minimal gaps in the interval 0–10. The last two digits are used to order all instances with the same characteristics.

In Tables 1 and 2 we present some of our results for graphs varying from 50 to 2000 vertices. The reduction rates are listed in both tables in the column *Reduction Rules*. Note that the number of optional edges is reduced, in some cases, in more than 90%. The objective function values obtained by both *BasedMYK* and *BasedLP* algorithms are presented, resp., in the columns *BasedMYK* and *BasedLP*. The best solution (*Best*(*MYK*+*LP*)-represented by boldface letters) is used as starting point in our TS procedure. Observe, for instance, the superior performance of the *BasedLP* for the majority of the instances we have considered. This can justify the use a more expensive LP solver in the BasedLP heuristic. However, if the size of the linear programming relaxation becomes prohibitive (due to a high execution time of the LP solver) we can use only the *BasedMYK* to produce starting points.

For instances between 50 and 500 vertices (see Tables 1 and 2), TS procedure was repeated 10 times in each case. The column *Best Value* shows the objective function value obtained at the best execution while the column *Average* exhibits the medium performance after all repetitions. The values between parentheses, indicate

Table 1
Results of the TS for instances with 50, 75 and 100 vertices.

Instance	Reduction rules (%)	Initial solution		Tabu search		Optimum	Time (s)
		BasedMYK	BasedLP	Best Value	Average		
G50-10-5-01	69.22	151	**176**	184 $_{(2)}$	181.05	185	12.28
G50-10-5-02	65.25	165	**198**	201 $_{(2)}$	198.04	207	5.29
G50-10-5-03	59.75	172	⋆ **268**	⋆ 268 $_{(10)}$	268	268	6.52
G50-10-5-04	65.38	136	**151**	160 $_{(2)}$	155.65	161	0.32
G50-10-5-05	59.57	151	**218**	218 $_{(10)}$	218	219	0.50
G75-15-7-01	81.04	374	**391**	415 $_{(1)}$	401.20	416	1.31
G75-15-7-02	51.40	372	⋆ **565**	⋆ 565 $_{(10)}$	565	565	18.40
G75-15-7-03	57.47	370	**509**	515 $_{(1)}$	509.55	521	0.83
G75-15-7-04	79.75	291	**297**	318 $_{(2)}$	304.05	320	13.09
G75-15-7-05	62.38	398	**535**	545 $_{(3)}$	539.15	548	0.96
G100-20-10-01	95.73	329	**363**	374 $_{(3)}$	364.45	379	35.21
G100-20-10-02	91.02	**360**	338	395 $_{(1)}$	385.18	401	8.94
G100-20-10-03	94.10	**354**	340	384 $_{(3)}$	377.78	388	24.68
G100-20-10-04	88.12	362	**420**	435 $_{(2)}$	422.50	439	2.97
G100-20-10-05	81.26	514	**578**	⋆ 602 $_{(2)}$	597.94	602	1.15

Table 2
Results of the TS for instances with 300, 500, 1000 and 2000 vertices.

Instance	Reduction rules (%)	Initial solution		Tabu		Path relinking		Approx.	Time (s)
		Based MYK	Based LP	Best value	Average	Best value	Average		
G500-50-5-1	61.93	9038	**11 408**	11 418 $_{(2)}$	11 417.2	–	–	0.9921	11.24
G500-50-5-2	62.42	9253	**11 793**	11 824 $_{(1)}$	11 819.5	–	–	0.9846	15.43
G500-50-5-3	60.23	9470	**12 622**	12 627 $_{(10)}$	12 627.0	–	–	0.9965	13.87
G500-50-5-4	60.18	8890	**12 028**	12 029 $_{(10)}$	12 029.0	–	–	0.9948	17.55
G500-50-5-5	61.67	9591	**12 235**	12 240 $_{(1)}$	12 239.1	–	–	0.9920	14.76
G1000-100-10-1	61.78	36 501	**46 777**	46 889 $_{(1)}$	46 849.9	46 951 $_{(2)}$	46 896.3	0.9892	95.36
G1000-100-10-2	59.96	37 065	**49 764**	49 942 $_{(3)}$	49 885.2	49 942 $_{(7)}$	49 923.1	0.9962	113.80
G1000-100-10-3	60.59	37 313	**48 574**	49 026 $_{(1)}$	48 946.8	49 027 $_{(1)}$	48 974.2	0.9938	113.79
G1000-100-10-4	61.20	38 192	**49 333**	49 509 $_{(1)}$	49 449.5	49 514 $_{(1)}$	49 473.2	0.9942	109.08
G1000-100-10-5	61.36	38 093	**48 752**	49 396 $_{(1)}$	49 229.6	49 413 $_{(1)}$	49 302.2	0.9906	101.81
G2000-200-20-1	61.31	**150 289**	–	184 365 $_{(1)}$	183 286.9	185 269 $_{(1)}$	183 943.8	0.9021	682.07
G2000-200-20-2	60.16	**146 317**	–	185 253 $_{(1)}$	184 752.0	185 464 $_{(1)}$	184 811.8	0.9162	685.19
G2000-200-20-3	60.93	**144 436**	–	179 641 $_{(1)}$	178 987.7	180 661 $_{(1)}$	179 325.5	0.9025	603.75
G2000-200-20-4	60.45	**146 641**	–	186 256 $_{(1)}$	184 593.0	186 430 $_{(1)}$	184 900.7	0.9114	697.38
G2000-200-20-5	60.50	**143 229**	–	181 872 $_{(1)}$	181 084.9	182 167 $_{(1)}$	181 358.7	0.9091	694.33

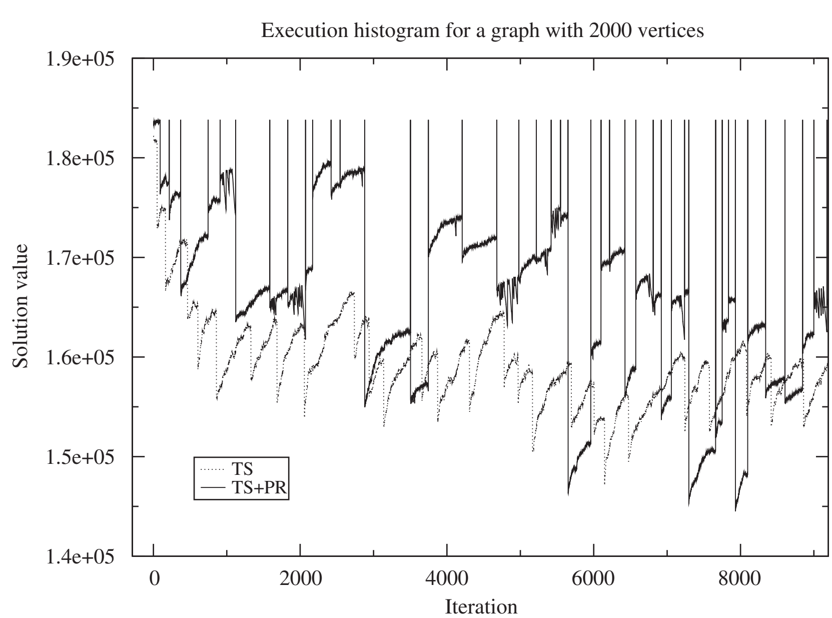

Fig. 4. Comparison between TS and TS+PR for an instance with 2000 vertices.

the number of times the best solution was found after 10 executions of the TS, and symbol ($\star$) in the first table, exhibits all those instances where the associated optimum values were encountered.

All parameters involved in the TS procedure were empirically defined and tested. The size of the tabu list was fixed around 10% of the total number of vertices belonging to $M_R \cup U_R$. In the diversification phase, as described in Section 4.1, we have chosen 10% (parameter k) of the total number of vertices f-controlled by M at a current solution G to be settled as non-f-controlled.

In our tests, the performance of the TS with PR was superior for some instances with up 1000 and 2000 vertices and they are listed in Table 2. For instances with less than 1000 vertices only the TS procedure was executed since we could not always guarantee an improvement after the PR execution (probably due to the good results of our TS for small instances). For instances with 1000 vertices, TS with PR was superior for only 10% of the tested instances and for instances with up 2000 vertices, the performance of the TS with PR was increased to 20%.

In Table 1, the optimum solution values are listed in the column *Optimum*. In the column *Approx.* of Table 2, for instances between 300 and 1000 vertices, we compute the approximation rates through the average performance ratio of the TS and the bounds gathered by the linear programming relaxation. Note, for instance, that for all instances considered the approximation rates were within 6% of the optimum value. For instances with 2000 vertices, these rates were worst, since they were computed using the maximum possible number of controlled vertices, i.e., $Z_H/(|V|-(|M_{NC}|+|U_{NC}|))$, with Z_H denoting the solution value obtained by our TS with PR procedure. Finally, the column *Time* in Tables 1 and 2, exhibits the worst execution time (at all repetitions) after the construction phase (in seconds), demanded by both TS and TS with PR.

We also illustrate the benefits of the combined TS with PR by executing a typical instance of the GMCSP with 2000 vertices (see Fig. 4). Note that the peaks in this graphic (corresponding to the execution of the TS with PR), denote the solution values obtained from a trajectory beginning at some arbitrary solution of elite set to the best current solution. The process is then restarted at the diversification step.

6. Conclusions

In this work, we presented a generalization of the max-controlled set problem (introduced by Makino et al. [17]). We presented a trivial $\frac{1}{2}$–approximation algorithm for the GMCSP and a new procedure for finding feasible solutions based on a strengthened linear programming formulation. These solutions were then used as starting points in our TS or TS with PR. Finally, we have presented some computational experiments comparing our results with the optimum solution values of the problem. Our tests indicated solutions within 6% of the optimal solutions. Further, the TS with PR introduced in this paper has demonstrated a superior performance when compared to the pure TS.

As a future direction, another intensification and diversification mechanisms may be considered through the use of frequency-based memory. Other possibility is to use PR to define new intensification strategies or combined with other classic metaheuristics such as GRASP, iterate local search, between others. Finally, as a consequence of Theorem 3, new randomized rounding procedures may be proposed for both MCSP and GMCSP by using our strengthened formulation (including the x,z variables) and by rounding the "z" variables (instead of the "x" variables).

Acknowledgments

We thank the CNPq/Brazil and FAPERJ/Brazil for their financial support and the anonymous referees for their insightful comments.

Appendix A. Proof of Theorem 3

Proof. Firstly, consider the following auxiliary notation. Given subsets $A,B \subseteq V$, we denote by $E_1(A,B)$, the set of fixed edges of $G_1=(V,E_1)$ with endpoints in A and B, respectively. If $A=\{v\}$ (or $B=\{v\}$) we just write v, for short.

Now suppose *w.l.o.g.*, that $V = M_R \cup U_R$. We conclude from (8) that an optimum fractional solution $(\overline{x},\overline{z})$ of $\overline{P}$ (with optimum value $\overline{z}_{max}$) must satisfy the following equation:

$$\frac{\sum_{v_i \in M} a_{ki}\overline{x}_{ki} - \sum_{v_j \in U} a_{kj}\overline{x}_{kj} - f_k}{b_k} + 1 = \overline{z}_k, \quad \forall v_k \in V \tag{11}$$

Note from (11) that, if v_k is f-controlled by M the solution $\overline{x}$ above satisfies the equation $\sum_{v_i \in M} a_{ki}\overline{x}_{ki} - \sum_{v_j \in U} a_{kj}\overline{x}_{kj} - f_k = 0$ and thus we can say that $\overline{z}_k \le 1$, $\forall v_k \in V$. In addition, from the definition of b_k (see Eq. (7)) we obtain $\overline{z}_k \ge 0$, $\forall v_k \in V$.

Now consider V' and E'_2, respectively, two sets of vertices and arcs as defined below:

$$V' = V \cup \{s,t\}$$
$$E'_2 = \{(v_i,v_j) | [v_i,v_j] \in E_2 \backslash E_1 \textit{ and } v_i \in M_R, v_j \in U_R\} \cup \{(s,v_i), \forall v_i \in M_R\} \cup \{(v_j,t), \quad \forall v_j \in U_R\}$$

Now, we construct an auxiliary network $N' = (G',c')$ with $G' = (V',E'_2)$ and non-negative arc capacities $c' : E'_2 \to \mathbb{R}^+$. The function c' will be constructed by using solution $\overline{z}$ and equality (11) in the following way:

$$c'_{si} = |E_1(v_i,M_R)| - |E_1(v_i,U_R)| - (\overline{z}_i - 1)b_i - f_i, \quad \forall v_i \in M_R$$
$$c'_{ij} = 1, \quad \forall v_i \in M_R \text{ and } v_j \in U_R$$
$$c'_{jt} = (\overline{z}_j - 1)b_j - |E_1(M_R,v_j)| + |E_1(U_R,v_j)| + f_j, \quad \forall v_j \in U_R \tag{12}$$

Note from (11) and (12), that $\overline{x}$ defines an optimum solution of the maximum flow problem between s and t in N' (since all arcs incident to s and t are saturated). Thus, by the flow conservation law at every vertex of V, we obtain

$$\sum_{\{v_j:(v_i,v_j) \in E'_2\}} a_{ij}\overline{x}_{ij} = \overline{x}_{si} = c'_{si}, \quad \forall v_i \in M_R \tag{13}$$

$$\sum_{\{v_i:(v_i,v_j) \in E'_2\}} a_{ij}\overline{x}_{ij} = \overline{x}_{jt} = c'_{jt}, \quad \forall v_j \in U_R \tag{14}$$

Now suppose, from our hypothesis, there exists at least one fractional variable $\overline{x}_{vu} \in (0,1)$ associated to $[v,u] \in E_2 \backslash E_1$. Obviously, $\overline{x}_{vu}$ also denotes the flow value in the arc $(v,u) \in E'_2$. Now we show how to construct a new optimum solution $(\hat{x},\hat{z})$ of $\overline{P}$ with component $\hat{x}_{vu} \in \{0,1\}$ and $\overline{z}_{max} = \sum_{v_i \in V} w_i\hat{z}_i$. Basically, the idea is to construct a network $N'' = (G',c'')$ with a new capacity function $c'' : E'_2 \to \mathbb{R}^+$ (conveniently defined) and solve the maximum flow problem between s and t in N'' using the *Augmenting Path Flow* algorithm (see [1]).

Initially consider $\delta' = \lfloor c'_{sv} \rfloor$ and $\delta'' = \lfloor c'_{ut} \rfloor$. In this case we have $\overline{x}_{vu} = \Delta \in (0,1)$, $\overline{x}_{sv} = \delta' + \Delta$ and $\overline{x}_{ut} = \delta'' + \Delta$, respectively. Note that we have Δ unities of flow through path $p=(s,v,u,t)$ in N' (where $v \in M_R$ and $u \in U_R$). In addition, assume that δ' and δ'' unities of

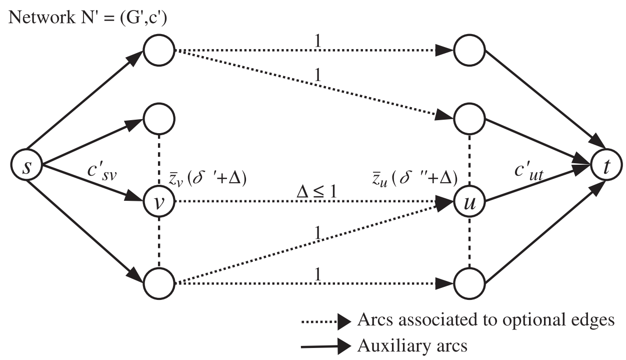

Fig. 5. Sending Δ units of flow in the path p=(s,v,u,t).

flow have been sent through others augmenting paths passing, respectively, by vertices v and u of V (see Fig. 5).

Now, we conclude from (11) that $\bar{z}_v = \bar{z}_v(\delta' + \Delta)$ and $\bar{z}_u = \bar{z}_u(\delta'' + \Delta)$ where

$$\bar{z}_v(\delta' + \Delta) = \frac{|E_1(v, M_R)| - (|E_1(v, U_R)| + (\Delta + \delta')) - f_v}{b_v} + 1, \quad \text{for } v \in M_R \tag{15}$$

$$\bar{z}_u(\delta'' + \Delta) = \frac{(|E_1(u, M_R)| + (\Delta + \delta'')) - |E_1(u, U_R)| - f_u}{b_u} + 1, \quad \text{for } u \in U_R \tag{16}$$

In the sequel, we show how to construct new capacities c''_{sv}, c''_{vu} and c''_{ut} at a new network $N'' = (G', c'')$ in such a way that, after solving the max-flow problem from s to t in N'', arcs (s,v), (u,t) are saturated and $w_v\bar{z}_v(\delta' + \Delta) + w_u\bar{z}_u(\delta'' + \Delta) = w_v\hat{z}_v(\delta' + \Delta') + w_u\hat{z}_u \quad (\delta'' + \Delta')$ where $\hat{z}_v = \hat{z}_v(\delta' + \Delta')$, $\hat{z}_u = \hat{z}_u(\delta'' + \Delta')$ and $\hat{x}_{vu} = \Delta' \in \{0, 1\}$. As a consequence of that, we obtain $\bar{z}_{max} = \hat{z}_{max}$. Note, for instance, from (15) and (16) that if it is possible to increase the flow in the arc (v,u) of N'' from Δ to $\Delta' = 1$, we always obtain $\hat{z}_v(\delta' + 1) \le \bar{z}_v(\delta' + \Delta)$ for $v \in M_R$ and $\hat{z}_u(\delta'' + 1) \ge \bar{z}_u(\delta'' + \Delta)$, for $u \in U_R$. Similarly, if we decrease the flow from Δ to $\Delta' = 0$ we obtain $\hat{z}_v(\delta') \ge \bar{z}_v(\delta' + \Delta)$ for $v \in M_R$ and $\hat{z}_u(\delta'') \le \bar{z}_u(\delta'' + \Delta)$ for $u \in U_R$.

Now we prove the following preliminary result: if $(\bar{x}, \bar{z})$ is an optimal fractional solution of $\overline{P}$ and $\bar{x}_{vu} = \Delta \in (0,1)$ for some edge $[v,u] \in E_2 \backslash E_1$ than $w_v/b_v = w_u/b_u$.

By contradiction, suppose $w_v/b_v \ne w_u/b_u$. Initially, consider $w_v/b_v < w_u/b_u$. In this case, we can define another network N'' with new capacities $c''_{sv} = \lfloor c'_{sv} \rfloor + 1$, $c''_{ut} = \lfloor c'_{ut} \rfloor + 1$ and $c''_{k\ell} = c'_{k\ell}$ for all the remaining arcs (v_k, v_ℓ) of N''. Solving the maximum flow problem between s and t in N'', we obtain an associated solution $(\hat{x}, \hat{z})$ of $\overline{P}$ with $\hat{x}_{vu} = 1$. Since $\Delta \in (0,1)$, we can prove from equality (11) above that $\bar{z}_v - \hat{z}_v = (1-\Delta)/b_v > 0$ and $\hat{z}_u - \bar{z}_u = (1-\Delta)/b_u > 0$, respectively. Thus:

$$\begin{aligned} \frac{w_v}{b_v} < \frac{w_u}{b_u} &\Rightarrow w_u\left(\frac{1-\Delta}{b_u}\right) - w_v\left(\frac{1-\Delta}{b_v}\right) > 0 \\ &\Rightarrow w_u(\hat{z}_u - \bar{z}_u) > w_v(\bar{z}_v - \hat{z}_v) \end{aligned} \tag{17}$$

Therefore, we can obtain a new solution $(\hat{x}, \hat{z})$ of $\overline{P}$ with $\hat{z}_k = \bar{z}_k$, $\forall v_k \in V \backslash \{v, u\}$ and $w_v\hat{z}_v + w_u\hat{z}_u > w_v\bar{z}_v + w_u\bar{z}_u$. Thus, we have obtained a new solution $(\hat{x}, \hat{z})$ of $\overline{P}$ with $\hat{z}_{max} > \bar{z}_{max}$, which is a contradiction.

Now consider $w_v/b_v > w_u/b_u$. Similarly, we can construct another network N'' with new capacities $c''_{sv} = \lfloor c'_{sv} \rfloor$, $c''_{ut} = \lfloor c'_{ut} \rfloor$ and $c''_{k\ell} = c'_{k\ell}$ for all the remaining arcs (v_k, v_ℓ) of N''. Solving the maximum flow problem between s and t, we obtain an associated solution $(\hat{x}, \hat{z})$ of $\overline{P}$ with $\hat{x}_{vu} = 0$. Since $\Delta \in (0,1)$, from Eq. (11) above we prove that $\hat{z}_v - \bar{z}_v = \Delta/b_v > 0$ and $\bar{z}_u - \hat{z}_u = \Delta/b_u > 0$, respectively. Therefore:

$$\begin{aligned} \frac{w_v}{b_v} > \frac{w_u}{b_u} &\Rightarrow w_v\left(\frac{\Delta}{b_v}\right) - w_u\left(\frac{\Delta}{b_u}\right) > 0 \\ &\Rightarrow w_v(\hat{z}_v - \bar{z}_v) > w_u(\bar{z}_u - \hat{z}_u) \end{aligned} \tag{18}$$

Therefore, we have obtained a new solution $(\hat{x}, \hat{z})$ of $\overline{P}$ with $\hat{z}_k = \bar{z}_k$, $\forall v_k \in V \backslash \{v, u\}$ and $w_v\hat{z}_v + w_u\hat{z}_u > w_v\bar{z}_v + w_u\bar{z}_u$. As a consequence of that we obtain $\hat{z}_{max} > \bar{z}_{max}$, resulting in a contradiction.

Finally consider $\Delta \in (0,1)$ and $w_v/b_v = w_u/b_u$. In this case, we can choose $\hat{x}_{vu} = \Delta' = 1$ or 0 at random and obtain a new network N'' with $c''_{sv} = \lfloor c'_{sv} \rfloor + \Delta'$, $c''_{ut} = \lfloor c'_{ut} \rfloor + \Delta'$ and $c''_{k\ell} = c'_{k\ell}$ for the remaining arcs (v_k, v_ℓ) of N''. Further, note from (11) that $\bar{z}_v = \hat{z}_v$ and $\bar{z}_u = \hat{z}_u$. Then, after solving the maximum flow problem between s and t in N'' we obtain an associated solution $(\hat{x}, \hat{z})$ of $\overline{P}$ with $\hat{x}_{vu} \in \{0, 1\}$ and $\sum_{v_i \in V} w_i\hat{z}_i = \sum_{v_i \in V} w_i\bar{z}_i$.

Repeating this process for every optional edge $[v_i, u_j]$ of G with $\hat{x}_{ij} \in (0,1)$ we obtain a final solution $(\tilde{x}, \tilde{z})$ of $\overline{P}$ with $\sum_{v_i \in V} w_i\tilde{z}_i = \sum_{v_i \in V} w_i\bar{z}_i$ and $\tilde{x}_{ij} \in \{0,1\}$, $\forall v_i \in M_R$ and $v_j \in U_R$. □

References

[1] Ahuja RN, Magnanti TL, Orlin JB. Network flows: theory, algorithms and applications. Englewoods Cliffs, NJ: Prentice-Hall; 1993.
[2] Bastos MP, Ribeiro CC. Reactive tabu search with path-relinking for the Steiner problem in graphs. In: Ribeiro CC, Hansen P, editors. Essays and surveys in metaheuristics. Dordrecht: Kluwer Academic Publishers; 2001. p. 39–58.
[3] Bermond JC, Bond J, Peleg D, Perennes S. The power of small coalitions in graphs. Discrete Applied Mathematics 2003;127:399–414.
[4] Carrano AV. Establishing the order of human chromosome-specific DNA fragments. In: Woodhead AD, Barnhart BJ, editors. Biotechnology and the human genome. New York: Plenum Press; 1988. p. 37–50.
[5] Fitoussi D, Tennenholtz V. Minimal social laws. In: Proceedings of the AAAI'98, 1998. p. 26–31.
[6] Glover F. Tabu search–part I. ORSA Journal on Computing 1989;1(3):190–206.
[7] Glover F, Laguna M. Modern heuristic techniques for combinatorial problems—tabu search. Oxford: Blackwell Scientific Publications; 1993. p. 70–150.
[8] Glover F. Tabu search and adaptive memory programming—advances applications and challenges. In: Barr R, Helgason R, Kennington J, editors. Interfaces in computer science and operations research. Dordrecht: Kluwer Academic Publishers; 1997. p. 1–75.
[9] Glover F, Laguna M. Tabu search. Dordrecht: Kluwer Academic Publishers; 1998.
[10] Glover F, Laguna M, Marti R. Fundamentals of scatter search and path relinking. Control and Cybernetics 2000;39(3):653–84.
[11] Golumbic MC, Kaplan H, Shamir R. On the complexity of DNA physical mapping. Advances in Applied Mathematics 1994;15:251–61.
[12] Golumbic MC, Kaplan H, Shamir R. Graph sandwich problems. Journal of Algorithms 1995;19(3):449–73.
[13] Hansen P. The steepest ascent mildest descent heuristic for combinatorial programming. In: Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy, 1986.
[14] Hassin Y, Peleg D. Distributed probabilistic polling and applications to proportionate agreement. Information and Computation 2001;171(2):248–68.
[15] Ho SC, Gendreau M. Path relinking for the vehicle routing problem. Journal of Heuristics 2006;12:55–72.
[16] Kaplan H, Shamir R. Bounded degrees interval sandwich problems. Algorithmica 1999;24:96–104.
[17] Makino K, Yamashita M, Kameda T. Max-and min-neighborhood monopolies. Algorithmica 2002;34:240–60.
[18] Martinhon CA, Protti F. An improved derandomized approximation algorithm for the max-controlled set problem. In: WEA 2004. Lecture notes in computer science, vol. 3059, Springer, Berlin, 2004. p. 341–55.
[19] Peleg D. Size bounds for dynamic monopolies. Discrete Applied Mathematics 1998;86(2–3):263–73.
[20] Peleg D. Local majorities, coalitions and monopolies in graphs: a review. Theoretical Computer Science 2002;282(2):231–57.
[21] Rose JD. A graph-theoretic study of the numerical solution of sparce positive definite systems of linear equations. In: Graph theory and computing. New York: Academic Press; 1972. p. 183–217.
[22] Santos IM. Approximation algorithms for the max-controlled set problem. Master Dissertation, Fluminense Federal University; 2005 [in Portuguese].
[23] Santos IM, Martinhon CA, Ochi LS. A VNS metaheuristic for the max-controlled set problem. Encontro Regional de Matemática Aplicada e Computacional-ERMAC, vol. 1, Rio de Janeiro, 2004. p. 30.
[24] Yannakakis IM. Computing the minimum fill-in is NP-complete. SIAM Journal on Algebraic and Discrete Methods 1981;2:77–9.