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

Ivairton M. Santos\textsuperscript{a}, Carlos A. Martinhon\textsuperscript{b,*}, Luiz S. Ochi\textsuperscript{b}

\textsuperscript{a} Mato Grosso Federal University, Médio Araguaia Institute, Rodovia MT-100, Km 3.5, Campus Universitário 1, 78698-000 Pontal do Araguaia, MT, Brazil
\textsuperscript{b} Fluminense Federal University, Department of Computer Science, Rua Passo da Pátria, 156 - Bloco E - 3 andar - Boa Viagem, 24210-240 Niterói, RJ, Brazil

A R T I C L E   I N F O

Available online 2 February 2010

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

A B S T R A C T

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 (GMCSPE), 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 \(G\) such that the sum of the weights associated to all vertices \(f\)-controlled by \(M\). We present a \(1\frac{1}{2}\)-approximation algorithm for the GMCSPE 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.

© 2010 Elsevier Ltd. All rights reserved.

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 \(P\) if \(G=(V,E)\) satisfies \(P\). A sandwich problem consists of deciding whether there exists some sandwich graph satisfying \(P\). We call fixed edges and optional edges, respectively, the edges belonging to \(E_1\) and \(E_2 \setminus E_1\). Many different properties may be considered in the context of sandwich graphs as, for example, physical mapping of DNA, temporal reasoning, synchronizing parallel processes, phylogenetic 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 (GMCSPE), 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...
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 $E$. 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 $1$-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 $|N_2(v_i) \cap M| = |N_1(v_i) \cap U| \geq f_i$, where $U = V \setminus M$. 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 $M$. The GMCSP is to find a sandwich graph $G$. We denote by $\text{minimum gap} (f_i, v_i)$ the value $f_i$ for which $v_i \in V$.

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 MSCP (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 $1$-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 $D(A,B) = \{v_i \mid (v_i) \in E_2 \cap E_1 | v_i \in A, v_j \in B\}$ 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 $E_3$ 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 $E_2$. Therefore, the set $E$ in the sandwich graph $G$ must satisfy: $E_1 \cup D(M,M) \subseteq E_2 \cup D(U,M)$ and $D(U,M) \cup D(U,U)$. For simplicity, assume from now on $E_1 = E_1$ (Reduction Rule 1) and $E_2 = E_2$ (Reduction Rule 2).

Prior to describing the following reduction rules, consider the following partition of $V$: we denote by $M_{NC}$ and $U_{NC}$, 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 $M_{RC}$ and $U_{RC}$ the subset of vertices which are never $f$-controlled by $M$ in any sandwich graph. Finally, we define the subsets $M_R = M \setminus (M_{NC} \cup M_{RC})$ and $U_R = U \setminus (U_{NC} \cup U_{RC})$. 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 $E = E_1$ we identify all vertices belonging to $M_{NC} \cup U_{NC}$. Similarly, if we set $E = E_2$, we determine $U_{NC} \cup M_{NC}$. The reduction rules are listed in the sequel.

- Add to $E_1$ all edges belonging to $D(M_{NC} \cup M_{RC}, U_R)$ (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 $D(M_{NC} \cup M_{RC}, U_{RC} \cup U_{NC})$ (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).
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 \( G^g(V,E) \) for \( E=E_1 \) and \( E=E_2 \) respectively. The variable \( z_{HI} \) 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. \( z_{HI} \leftarrow \max(W_1, W_2) \);

**Theorem 1.** Let \( z_{opt} \) be the optimum solution value of the GMCSP.
Then \( z_{HI} \geq \frac{1}{2} z_{opt} \).

**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 \( z_{ HI} = \max(W_1, W_2) \) it follows that \( z_{HI} \geq \frac{1}{2} z_{max} \). \( \square \)

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 \( E_2 \) 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 \( v_i \). The constants \( a_{ij} \in \{0,1\} \) are associated to \([v_i, v_j] \) 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 \( K = |U| + \max(f_j, \forall v_j \in V) \) for \( U = V \setminus M \). Then, we have the following linear programming formulation for the GMCSP:

\[
\begin{align*}
z_{max} = \text{Maximize} \sum_{v_i \in V} w_i z_i \\
\text{subject to:} \\
\sum_{v_j \in M} a_{ij} x_{ij} - \sum_{v_j \in U} a_{ij} x_{ij} - f_i \leq K z_i, & \quad \forall v_i \in V \\
x_j = 1, \quad \forall [v_i, v_j] \in E_1 \\
x_j = 0, \quad \forall v_j \in V \\
x_j \in \{0,1\}, \quad \forall [v_i, v_j] \in E_2 \setminus E_1 \\
z_i \in \{0,1\}, \quad \forall v_i \in V
\end{align*}
\]

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 \( M_R, M_AC, U_R, U_AC, U_NC \) 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:

\[
b_i = \sum_{[v_j, v_j] \in \tilde{Z}} a_{ij} x_{ij} - \sum_{v_j \in U} a_{ij} x_{ij} - f_i \quad \text{for} \quad v_i \in M_R \cup U_R
\]

The constant \( b_i \) is computed in the following way: if \( v_i \in M_R \), we set \( x_j = 1, \forall [v_i, v_j] \in E_2 \setminus E_1 \). Analogously, if \( v_i \in U_R \), we set \( x_j = 0, \forall [v_i, v_j] \in E_2 \setminus E_1 \). Obviously, in both cases we must have \( x_j = 1, \forall [v_i, v_j] \in E_1 \). Thus, a stronger formulation for the GMCSP can be obtained by changing constraint (2) by constraints (8)–(10) as described in the sequel:

\[
\begin{align*}
\sum_{v_j \in M} a_{ij} x_{ij} - \sum_{v_j \in U} a_{ij} x_{ij} - f_i \leq K z_i, & \quad \forall v_i \in M_R \cup U_R \\
z_i = 1, \quad \forall v_i \in M.AC \cup U.AC \\
z_i = 0, \quad \forall v_i \in M.NC \cup U.NC
\end{align*}
\]
Let $\mathcal{P}$ 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 $z_{\text{max}}$ and $\mathcal{Z}_{\text{max}}$ be the optimum values of $\hat{P}$ and $\mathcal{P}$, respectively. Then, $z_{\text{max}} \geq \mathcal{Z}_{\text{max}}$.

**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 $(\hat{x}, z)$ be a feasible solution of both $\hat{P}$ and $\mathcal{P}$, respectively. If vertex $v_i$ is $f$-controlled by $M$ at some integer feasible solution than both $z_i = z = 1$ (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 $z_i \geq z = 1$, $\forall v_i \in V$. Therefore $\sum_{v_i \in V} w_i z_i \geq \sum_{v_i \in V} w_i z = \sum_{v_i \in V} w_i z_i$.

It is assumed from now on that $(\mathcal{X}, \mathcal{Z})$ will denote an optimal solution of the strengthened formulation $\mathcal{P}$ and $z_{\text{max}}$ its associated objective function value. The optimum solution value of the GMCS is denoted by $z_{\text{max}}$.

A feasible solution for the GMCS can be constructed through the linear relaxation in the following way. Given $(\mathcal{X}, \mathcal{Z})$ of $\mathcal{P}$ with components $\mathcal{X}_{\mathcal{E}} \in [0, 1], \forall \mathcal{E} \in \mathcal{E}_2$ and $\mathcal{Z}_{\mathcal{E}} \in [0, 1], \forall \mathcal{E} \in V$, we define as $f$-controlled all vertices $v_i \in V$ with $z_i = 1$, and as not $f$-controlled (or uncontrolled) the remaining vertices of $V$ with $z_i < 1$. However, to decide about the set of optimal edges with $\mathcal{X}_{\mathcal{E}} \in (0, 1)$? Actually, the following result ensure that binary values $\mathcal{X}_{\mathcal{E}} \in (0, 1)$ are always obtained after solving the linear relaxation $\mathcal{P}$. 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 optimal edges will be selected or not in our BaseLP heuristic. Formally, we have established the following result (proved in the Appendix):

**Theorem 3.** Consider $(\mathcal{X}, \mathcal{Z})$ an optimum solution of $\mathcal{P}$ with components $0 \leq \mathcal{X}_{\mathcal{E}} \leq 1, \forall \mathcal{E} \in \mathcal{E}_2, \mathcal{Z}_{\mathcal{E}} \in [0, 1], \forall \mathcal{E} \in V$ and optimum value $z_{\text{max}} = \sum_{v_i \in V} w_i z_i$. If $0 < \mathcal{X}_{\mathcal{E}} < 1$ for some edge $\mathcal{E} = (v_i, v_j) \in \mathcal{E}_1$, we have another feasible solution $(\mathcal{X}, \mathcal{Z})$ of $\mathcal{P}$ with $z_{\text{max}}' = \sum_{v_i \in V} w_i z_i'$, where $\mathcal{X}_{\mathcal{E}} = \min(1, \mathcal{X}_{\mathcal{E}})$ and $\mathcal{Z}_{\mathcal{E}} \in [0, 1], \forall \mathcal{E} \in V$.

Theorem 3 above is illustrated in the example of Fig. 3. Consider $w_i = w_j$ and $f = f_0 = 0, \forall v_i, v_j \in V$. In both cases, we have the objective function value 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 the fractional values defining probabilities are only possible for instances with $w_i/b_i = w_j/b_j$ and $[\mathcal{X}_{\mathcal{E}} \in \mathcal{E}_1]$. 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 GMCS) for $f_0 = 0$ and $w_i = 1, \forall v_i \in V$, the Best(MYK+LP) procedure has a performance ratio equal to $(1/2) + (1 + \sqrt{1/2})/(n - 1)$, $n > 4$. In [18], the authors propose a randomized rounding and a derandomization procedure for the MCSP using probabilities defined by the linear relaxation of $\hat{P}$ in their formulation $w_i = 1, f_0 = 0$ and $b_i = K, w_i \in V$. Thus, as a consequence of Theorem 3 (when restricted to MCSP), the derandomization step proposed in [18] is not necessary, since after solving $\hat{P}$ by the simplex algorithm, we always obtain 0–1 assignments for all coordinates of vertex $x$.

4. A Tabu Search with Path Relinking for the GMCS

Given a function $h(\cdot)$ to be maximized over a set $\mathcal{P}$ of feasible solutions, Tabu Search-TS procedure starts from an initial point in $\mathcal{P}$ and proceeds iteratively from one point in $\mathcal{P}$ to another until some stop condition is met. To each solution \$ \in \mathcal{P}, one associates a set of neighbors $N(S) \in \mathcal{P}$. 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 $S$ of selected solutions. The PR was first introduced in [7] and can be viewed as a strategy that seeks to incorporate attributes of $S$ 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 $G = (V, \mathcal{E})$ (associated to a current solution $S$), we denote by $N_G(S)$, 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 by the local search.

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

![Fig. 3. Two optimal solutions with fractional and integer coordinates, respectively.](image-url)
As discussed before, given a sandwich graph $G$, our goal in the neighborhood $N_f(G)$ is to $f$-control new vertices in $L_0$ maintaining all vertices in $L_C$ (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 $L_0 \neq \emptyset$ do
3: Pick at random $v_i \in L_0$
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: $v_i \leftarrow$ Randomly choose a vertex in $H(v_i)$
8: if $v_i \notin M_B$ then
9: $E = E - \{v_i, v_j\}$
10: else
11: $E = E \cup \{v_i, v_j\}$
12: end if
13: $f_G(v_i) = f_G(v_i) + 1$
14: $f_G(v_i) = f_G(v_i) - 1$
15: $H(v_i) = H(v_i) \backslash v_j$
16: end while
17: end if
18: end if
19: $L_0 = L_0 \cup \{v_i\}$
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 $L_0$ 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 $N_f(G)$, 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 $v_i \in L_0$. In addition, from lines 6–16, note that if a vertex $v_i$ to be $f$-controlled by $M$ belongs to $L_0 \cap M_B$ (respectively $L_0 \cap U_B$) 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 $L_0$ (lines 15 and 19). Note, for instance, that all vertices $v_i \in H(v_i)$ remains $f$-controlled or non-$f$-controlled by $M$ after each updating.

The tabu list $T$ (subset of $L_0$) 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 $L_0$. If $v_i \in M_C$ (respectively $v_i \in U_C$) 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 $N_f(G)$ for $f \geq 1$). Given a sandwich graph $G$, the local search may be performed, for example, by removing $\ell$ 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 $N_f(G)$ [23].

We combine TS and PR by trying to incorporate good attributes of our best solutions (elite set $E$). 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 $E$. Parameters $l_{max}$ 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 $E$, and in the lines 19–20 we update the set of f-controlled vertices $L_0$ 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: $G^* \leftarrow G$, $z_{f_G}(G^*) \leftarrow \infty$, Initialize elite set $E \leftarrow \emptyset$
3: $i \leftarrow 0$, $j \leftarrow 0$ {Auxiliary variables}
4: while $i < l_{max}$ do
5: $T \leftarrow \emptyset$ {Initialize the tabu list}
6: while $j < j_{max}$ do
7: $G^* \leftarrow \text{LocalSearch}(N_f(G) \cap T)$ through Aspiration Criteria
8: $G^* \leftarrow \text{LocalSearch}(N_f(G) \cap T)$ through Aspiration Criteria
9: if $z_{f_G}(G^*) > z_{f_G}(G)$ then
10: $G^* \leftarrow G^*$
11: end if
12: if $z_{f_G}(G) > z_{f_G}(G^*)$ then
13: $G^* \leftarrow G$
14: $j \leftarrow 0$
15: end if
16: if $z_{f_G}(G) > \text{"worst solution in the elite set } E$ then
17: Add $G^*$ to the elite set $E$
18: end if
19: Choose, at random, a subset $P \subseteq L_0$
20: Update subsets $L_0 \leftarrow L_0 \backslash P$ and $T \leftarrow T \cup P$
21: $j \leftarrow j + 1$
22: end while
23: $G^* \leftarrow \text{Path Relinking}(E)$
24: $G^* \leftarrow \text{Diversification}(G^*)$
25: $i \leftarrow i + 1$
26: end while
27: Return $G^*$

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 $L_0 = M_C \cup U_C$) 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 $P \subseteq M_C$ or $P \subseteq U_C$, otherwise, if $P$ contains vertices of both $M_C$ and $U_C$, the simultaneous removal of all vertices of $P$ could be more complicated and computationally expensive. Thus, to change the status of a single vertex $v \in M_C$ (resp., $v \in U_C$) 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 $E$ and $T$, and the number of iterations $l_{max}$ 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 \( \| \)). 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 \( \| \), we consider two sandwich graphs \( G_{\text{source}} \) and \( G_{\text{dest}} \) (resp., \( \text{source} \) and \( \text{destination} \)) belonging to \( \| \) with the biggest diversity between them (i.e., number of distinct components of A).

Therefore, if \( A_{\text{source}} \) and \( A_{\text{dest}} \) are two associated vectors of binary 0–1 values, we hope to find better solutions by following a trajectory from \( A_{\text{source}} \) and \( A_{\text{dest}} \) (equivalently between \( G_{\text{source}} \) and \( G_{\text{dest}} \)) 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 \( G_{\text{source}} \) and \( G_{\text{dest}} \) construct attributes \( A_{\text{source}} \) and \( A_{\text{dest}} \)
2: \( R \leftarrow \) set of the different attributes between \( A_{\text{source}} \) and \( A_{\text{dest}} \)
3: while \( R \neq \emptyset \) do
4: \( A, A' \leftarrow A_{\text{source}}, A_{\text{dest}} \)
5: for every \( i \in R \) do
6: \( A'[i] \leftarrow A_{\text{dest}}[i] \)
7: if \( \text{cost}(A) \geq \text{cost}(A') \) then
8: \( A' \leftarrow A' \) and \( j \leftarrow i \)
9: end if
10: \( A'[i] \leftarrow A_{\text{source}}[i] \)
11: end for
12: \( A_{\text{source}} \leftarrow A' \)
13: if \( \text{cost}(A_{\text{source}}) > \text{cost}(A') \) then
14: \( A' \leftarrow A_{\text{source}} \)
15: end if
16: \( R \leftarrow R \setminus \{j\} \)
17: end while
18: Return \( G' \)

In the line 1, we construct attribute vectors \( A_{\text{source}} \) and \( A_{\text{dest}} \) associated respectively to \( G_{\text{source}} \) and \( G_{\text{dest}} \). 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 \( A_{\text{dest}} \). This may be accomplished in the following way.

Firstly, in the line 6, we create a new temporary solution \( A'' \) with attribute \( A'[i] = A_{\text{dest}}[i] \). 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 \( G_{\text{dest}} \). 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 \( A_{\text{source}} \) and in the lines 13, 14 we update the best attribute vector \( A'' \). 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 \( G' \) associated to \( A'' \).

In our tests we have settled \( G_{\text{dest}} = G' \), \( G_{\text{source}} = G \) (arbitrarily chosen at \( \| \)) and \( |\| = 10 \). Different combinations of our targets and \( |\| \) 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 \( E_z \) 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 median performance after all repetitions. The values between parentheses, indicate
Table 1
Results of the TS for instances with 50, 75 and 100 vertices.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Reduction rules (%)</th>
<th>Initial solution</th>
<th>Tabu search</th>
<th>Optimum</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Based MYK</td>
<td>Based LP</td>
<td>Best Value</td>
<td>Average</td>
</tr>
<tr>
<td>G50-10-5-01</td>
<td>69.22</td>
<td>151</td>
<td>176</td>
<td>184 (2)</td>
<td>181.05</td>
</tr>
<tr>
<td>G50-10-5-02</td>
<td>65.25</td>
<td>165</td>
<td>198</td>
<td>201 (2)</td>
<td>198.04</td>
</tr>
<tr>
<td>G50-10-5-03</td>
<td>59.75</td>
<td>172</td>
<td>151</td>
<td>160 (2)</td>
<td>155.65</td>
</tr>
<tr>
<td>G50-10-5-04</td>
<td>65.38</td>
<td>136</td>
<td>151</td>
<td>160 (2)</td>
<td>155.65</td>
</tr>
<tr>
<td>G50-10-5-05</td>
<td>59.57</td>
<td>151</td>
<td>218</td>
<td>218 (10)</td>
<td>218</td>
</tr>
<tr>
<td>G75-15-7-01</td>
<td>81.04</td>
<td>374</td>
<td>391</td>
<td>415 (1)</td>
<td>401.20</td>
</tr>
<tr>
<td>G75-15-7-02</td>
<td>51.40</td>
<td>372</td>
<td>565</td>
<td>565 (10)</td>
<td>565</td>
</tr>
<tr>
<td>G75-15-7-03</td>
<td>57.47</td>
<td>370</td>
<td>509</td>
<td>515 (1)</td>
<td>509.55</td>
</tr>
<tr>
<td>G75-15-7-04</td>
<td>79.75</td>
<td>291</td>
<td>297</td>
<td>318 (2)</td>
<td>304.05</td>
</tr>
<tr>
<td>G75-15-7-05</td>
<td>62.38</td>
<td>398</td>
<td>533</td>
<td>545 (3)</td>
<td>539.15</td>
</tr>
<tr>
<td>G100-20-10-01</td>
<td>95.73</td>
<td>329</td>
<td>363</td>
<td>374 (3)</td>
<td>364.45</td>
</tr>
<tr>
<td>G100-20-10-02</td>
<td>91.02</td>
<td>360</td>
<td>338</td>
<td>395 (1)</td>
<td>385.18</td>
</tr>
<tr>
<td>G100-20-10-03</td>
<td>94.10</td>
<td>354</td>
<td>340</td>
<td>384 (3)</td>
<td>377.78</td>
</tr>
<tr>
<td>G100-20-10-04</td>
<td>88.12</td>
<td>362</td>
<td>420</td>
<td>435 (2)</td>
<td>422.50</td>
</tr>
<tr>
<td>G100-20-10-05</td>
<td>81.26</td>
<td>514</td>
<td>578</td>
<td>602 (2)</td>
<td>597.94</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Instance</th>
<th>Reduction rules (%)</th>
<th>Initial solution</th>
<th>Tabu</th>
<th>Path relinking</th>
<th>Approx.</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Based MYK</td>
<td>Based LP</td>
<td>Best value</td>
<td>Average</td>
<td>Best value</td>
</tr>
<tr>
<td>G500-50-5-1</td>
<td>61.93</td>
<td>9038</td>
<td>11408</td>
<td>11418 (1)</td>
<td>11417.2</td>
<td>–</td>
</tr>
<tr>
<td>G500-50-5-2</td>
<td>62.42</td>
<td>9253</td>
<td>11793</td>
<td>11824 (1)</td>
<td>11819.5</td>
<td>–</td>
</tr>
<tr>
<td>G500-50-5-3</td>
<td>60.23</td>
<td>9470</td>
<td>12622</td>
<td>12627 (10)</td>
<td>12627.0</td>
<td>–</td>
</tr>
<tr>
<td>G500-50-5-4</td>
<td>60.18</td>
<td>8890</td>
<td>12028</td>
<td>12029 (10)</td>
<td>12029.0</td>
<td>–</td>
</tr>
<tr>
<td>G500-50-5-5</td>
<td>61.67</td>
<td>9951</td>
<td>12235</td>
<td>12240 (1)</td>
<td>12239.1</td>
<td>–</td>
</tr>
<tr>
<td>G1000-100-10-1</td>
<td>61.78</td>
<td>36501</td>
<td>46777</td>
<td>46889 (1)</td>
<td>46849.9</td>
<td>46951 (2)</td>
</tr>
<tr>
<td>G1000-100-10-2</td>
<td>59.96</td>
<td>37065</td>
<td>49764</td>
<td>49942 (1)</td>
<td>49885.2</td>
<td>49942 (7)</td>
</tr>
<tr>
<td>G1000-100-10-3</td>
<td>60.59</td>
<td>37313</td>
<td>48574</td>
<td>49026 (1)</td>
<td>48946.8</td>
<td>49027 (1)</td>
</tr>
<tr>
<td>G1000-100-10-4</td>
<td>61.20</td>
<td>38192</td>
<td>49333</td>
<td>49509 (1)</td>
<td>49449.5</td>
<td>49514 (1)</td>
</tr>
<tr>
<td>G1000-100-10-5</td>
<td>61.36</td>
<td>38093</td>
<td>48752</td>
<td>49396 (1)</td>
<td>49229.6</td>
<td>49413 (1)</td>
</tr>
<tr>
<td>G2000-200-20-1</td>
<td>61.31</td>
<td>150289</td>
<td>–</td>
<td>184365 (1)</td>
<td>183286.9</td>
<td>185269 (1)</td>
</tr>
<tr>
<td>G2000-200-20-2</td>
<td>60.16</td>
<td>146317</td>
<td>–</td>
<td>185253 (1)</td>
<td>184752.0</td>
<td>185464 (1)</td>
</tr>
<tr>
<td>G2000-200-20-3</td>
<td>60.93</td>
<td>144436</td>
<td>–</td>
<td>179641 (1)</td>
<td>178978.7</td>
<td>180661 (1)</td>
</tr>
<tr>
<td>G2000-200-20-4</td>
<td>60.45</td>
<td>146641</td>
<td>–</td>
<td>186236 (1)</td>
<td>184593.0</td>
<td>186430 (1)</td>
</tr>
<tr>
<td>G2000-200-20-5</td>
<td>60.50</td>
<td>143229</td>
<td>–</td>
<td>181872 (1)</td>
<td>181084.9</td>
<td>182167 (1)</td>
</tr>
</tbody>
</table>

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 (*) 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 $M_g \cup U_g$. 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 to 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 to 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., $\overline{Z}_{bat}/(|V|-|MN_c|+|U_R|)$, with $\overline{Z}_{bat}$ 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.

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 optimum 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 $x_2$ 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 $E_i(A, B)$ the set of fixed edges of $G_i(V, E_i)$ 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 $V=M_g \cup U_R$. We conclude from (8) that an optimum fractional solution $(x, y)$ of $P$ (with optimum value $z_{\text{max}}$) must satisfy the following equation:

$$\sum_{v \in E_i} a_{vi} x_{vi} - \sum_{v \in E_i} \alpha_{vi} y_{vi} - f_k = 0 \quad \forall v \in V$$

(11)

Note from (11) that, if $v_k$ is $f$-controlled by $M$ the solution $x$ above satisfies the equation $\sum_{v \in E_i} a_{vi} x_{vi} - \sum_{v \in E_i} \alpha_{vi} y_{vi} - f_k = 0$ and thus we can say that $z_k \leq 1$, $\forall v_k \in V$. In addition, from the definition of $b_k$ (see Eq. (7)) we obtain $z_k, \forall v_k \in V$.

Now consider $V'$ and $E_2$, respectively, two sets of vertices and arcs as defined below:

$$V' = V \cup \{s, t\}$$

$$E_2 = \{(v, v') | (v, v') \in E_2, E_1 \text{ and } v \in M_g \cup U_R \cup \{(s, v), (v, t), (v', v) \in M_g \cup U_R \}$$

Now, we construct an auxiliary network $N' = (G', c')$ with $G' = (V', E_2)$ and non-negative arc capacities $c' : E_2 \rightarrow \mathbb{R}^+$. The function $c'$ will be constructed by using solution $z$ and equality (11) in the following way:

$$c'_s = \left|E_1(v, M_g)\right| - \left|E_1(v, U_R)\right| - \left|E_1(v, M_g \cup U_R)\right| + \left|E_1(U_R, v)\right| + \left|E_1(U_R, v)\right| + f_j, \quad \forall v \in M_R$$

(12)

Note from (11) and (12), that $x$ 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

$$\sum_{(v, u) \in E_2} a_{vu} y_{vu} = x_{vu} = c'_{vu}, \quad \forall v \in M_R$$

(13)

$$\sum_{(v, u) \in E_2} a_{vu} y_{vu} = x_{vu} = c'_{vu}, \quad \forall v \in U_R$$

(14)

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

Initially consider $\delta' = [c'_{vu}]$ and $\delta'' = [c''_{vu}]$. In this case we have $x_{vu} = \delta' = \delta'' \neq 1$ and $x_{vu} = \delta' + \delta''$, respectively. Note that we have $\delta$ (or $\delta''$) units 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 $\delta'$ (or $\delta''$) units of
Now consider $w_v/b_v > w_u/b_u$. Similarly, we can construct another network $N'$ with new capacities $c'_{vw} = |c_{vw}| + 1$, $c'_{vu} = |c_{vu}| + 1$ and $c'_{wu} = c_{wu}$ for all the remaining arcs $(v_u, v')$ of $N'$. Solving the maximum flow problem between $s$ and $t$ in $N'$, we obtain an associated solution $(\tilde{x}, \tilde{z})$ of $P$ with $\tilde{x}_{vu} = 0$. Since $A \in (0, 1)$, from Eq. (11) above we prove that $\tilde{z}_v - \tilde{z}_t = \Delta/b_v > 0$ and $\tilde{z}_u - \tilde{z}_t = \Delta/b_u > 0$, respectively. Therefore:

$$w_v/b_v > w_u/b_u \Rightarrow w_v (\Delta/b_v) - w_u (\Delta/b_u) > 0$$

$$\Rightarrow w_v (\tilde{z}_v - \tilde{z}_t) > w_u (\tilde{z}_u - \tilde{z}_t)$$

(17)

Therefore, we have obtained a new solution $(\tilde{x}, \tilde{z})$ of $P$ with $\tilde{z}_k = \tilde{z}_k$, $\forall v \epsilon V \setminus \{v, u\}$ and $w_v \tilde{z}_v + w_u \tilde{z}_u > w_v \tilde{z}_v + w_u \tilde{z}_u$. Thus, we have obtained a new solution $(\tilde{x}, \tilde{z})$ of $P$ with $\tilde{z}_{vu} > \tilde{z}_{vu}$, which is a contradiction.

Now consider $w_v/b_v > w_u/b_u$. Similarly, we can construct another network $N'$ with new capacities $c'_{vw} = |c_{vw}| + 1$, $c'_{vu} = |c_{vu}| + 1$ and $c'_{wu} = c_{wu}$ for all the remaining arcs $(v_u, v')$ of $N'$. Solving the maximum flow problem between $s$ and $t$, we obtain an associated solution $(\tilde{x}, \tilde{z})$ of $P$ with $\tilde{x}_{vu} = 0$. Since $A \in (0, 1)$, from Eq. (11) above we prove that $\tilde{z}_v - \tilde{z}_t = \Delta/b_v > 0$ and $\tilde{z}_u - \tilde{z}_t = \Delta/b_u > 0$, respectively. Therefore:

$$w_v/b_v > w_u/b_u \Rightarrow w_v (\Delta/b_v) - w_u (\Delta/b_u) > 0$$

$$\Rightarrow w_v (\tilde{z}_v - \tilde{z}_t) > w_u (\tilde{z}_u - \tilde{z}_t)$$

(18)

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

Finally consider $A = (0, 1)$ and $w_v/b_v = w_u/b_u$. In this case, we choose $\tilde{x}_v = 0$, $\tilde{z}_v = 0$ and $\tilde{z}_u = \tilde{z}_u$. Then, after solving the maximum flow problem between $s$ and $t$ in $N'$ we obtain an associated solution $(\tilde{x}, \tilde{z})$ of $P$ with $\tilde{x}_{vu} \epsilon (0, 1)$ and $\tilde{z}_{vu} = \tilde{x}_{vu}$. Repeating this process for every optional edge $[v, u]_t$ of $G$ with $x_t \in \{0, 1\}$ we obtain a final solution $(\tilde{x}, \tilde{z})$ of $P$ with $\sum_{v \epsilon V} \tilde{z}_{vu} = \tilde{z}_{vu}$ and $\tilde{x}_v \epsilon (0, 1)$, $\forall v \epsilon V \setminus \{v, u\}$.