Journal of Mathematical Sciences, Vol. 161, No. 6, 2009

A HEURISTIC FOR THE STABILITY NUMBER OF A GRAPH BASED ON CONVEX QUADRATIC PROGRAMMING AND TABU SEARCH

L. Cavique and C. J. Luz

UDC 519.853.32

Abstract. Recently, a characterization of the Lovász theta number based on convex quadratic programming was established. As a consequence of this formulation, we introduce a new upper bound on the stability number of a graph that slightly improves the theta number. Like this number, the new bound can be characterized as the minimum of a function whose values are the optimum values of convex quadratic programs. This paper is oriented mainly to the following question: how can the new bound be used to approximate the maximum stable set for large graphs? With this in mind we present a two-phase heuristic for the stability problem that begins by computing suboptimal solutions using the new bound definition. In the second phase a multi-start tabu search heuristic is implemented. The results of applying this heuristic to some DIMACS benchmark graphs are reported.

1. Introduction

Let $G = (V, E)$ be a simple undirected graph, where $V = \{1, \ldots, n\}$ denotes the vertex set and E is the edge set. It will be supposed that G has at least one edge, i.e., E is not empty. We will write $ij \in E$ to denote the edge linking nodes i and j of V. The adjacency matrix $A_G = [a_{ij}]$ of G is the symmetric matrix defined by

$$a_{ij} = \begin{cases} 1 & \text{if} \quad ij \in E, \\ 0 & \text{if} \quad ij \notin E. \end{cases}$$

A stable set (or independent set) of G is a subset of nodes of V whose elements are pairwise nonadjacent. The stability number (or independence number) of G is defined as the cardinality of a largest stable set and is usually denoted by $\alpha(G)$. A maximum stable set of G is a stable set with $\alpha(G)$ nodes.

A clique of G is a subset of nodes of V whose elements are pairwise adjacent. The clique number of G is the cardinality of a largest clique and is usually denoted by $\omega(G)$. A maximum clique of G is a clique with $\omega(G)$ nodes.

These definitions imply that $\omega(G) = \alpha(\overline{G})$, where $\overline{G}$ is the complement subgraph of G. Consequently, to determine the maximum clique of a graph one can look for the maximum stable set of its complement graph. Hence any algorithm solving one of these problems can also be applied to solve the other.

The problem of finding $\alpha(G)$ (or, equivalently, $\omega(G)$) is NP-hard and the same happens with the more general problem of determining the maximum stable set (or the maximum clique) of G. However, several ways to obtain approximative solutions of these problems have been proposed in the literature. One of them is given in [19], where the following convex quadratic upper bound υ on $\alpha(G)$ was introduced:

$$(P_G) \qquad \upsilon = \max\{2e^T x - x^T (H + I)\, x : \, x \geq 0\}, \tag{1}$$

where e is the $n \times 1$ all ones vector, T denotes for the transposition operation, I is the identity matrix of order n, and

$$H = \frac{1}{-\lambda_{\min}(A_G)} A_G.$$

Translated from Sovremennaya Matematika i Ee Prilozheniya (Contemporary Mathematics and Its Applications), Vol. 63, Optimal Control, 2009.

A_G denotes, as above, the adjacency matrix of G, and $\lambda_{\min}(A_G)$ is its smallest eigenvalue. Since the trace of A_G is zero and G has at least one edge, A_G is indefinite. Hence $\lambda_{\min}(H) = -1$ and this guarantees the convexity of (P_G) since $H + I$ is positive semidefinite.

The upper bound υ was generalized in [20], where the following family of quadratic problems, based on a perturbation in the Hessian of problem (P_G), was introduced:

$$(P_G(C)) \qquad \upsilon(C) = \max\{2e^T x - x^T (H_C + I)\, x : x \geq 0\},$$

where $C = [c_{ij}]$ is a nonzero real symmetric matrix such that $c_{ij} = 0$ if $i = j$ or $ij \notin E$, and

$$H_C = \frac{C}{-\lambda_{\min}(C)}.$$

Any matrix satisfying the conditions imposed on the matrix C is called a *weighted adjacency matrix* of G. Note that as well as the adjacency matrix A_G, the matrix C is indefinite taking into account that its trace is null and not all entries c_{ij} are null. Consequently, since $\lambda_{\min}(H_C) = -1$, all problems $(P_G(C))$ are convex. Note also that $\upsilon(A_G) = \upsilon$ and, therefore, (P_G) is included into the introduced family of quadratic problems.

We can easily assert that for any weighted adjacency matrix C of a graph G, the number $\upsilon(C)$ is an upper bound on $\alpha(G)$. In fact, to see this, let x be a characteristic vector of any maximum independent set S of G (defined by $x_i = 1$ if $i \in S$ and $x_i = 0$ otherwise). Since the vector x is a feasible solution of $(P_G(C))$ and satisfies $x^T H_C\, x = 0$ (note that $x_i x_j = 0$ if $ij \in E$), we have

$$\upsilon(C) \geq 2e^T x - x^T x - x^T H_C\, x = 2\alpha(G) - \alpha(G) = \alpha(G),$$

i.e., $\alpha(G) \leq \upsilon(C)$, for all weighted adjacency matrices C of G.

As is proved in [20], the best upper bound $\upsilon(C)$ coincides with the Lovász theta number, i.e.,

$$\vartheta(G) = \min_C \upsilon(C) = \min_C \max\left\{2e^T x - x^T (H_C + I)x : x \geq 0\right\},$$

where C is a weighted adjacency matrix of G. The Lovász theta number was introduced in [18], is computable in polynomial-time and has been subsequently studied in several publications. It is the most famous upper bound of $\alpha(G)$ generally considered, for which several different formulations were established in the literature (see [13, 17]).

Now we introduce a slightly more accurate upper bound on $\alpha(G)$, defined as follows:

$$\vartheta^-(G) = \min_C \upsilon^-(C), \tag{2}$$

where C is a weighted adjacency matrix and

$$(P_G^-(C)) \qquad \upsilon^-(C) = \max\{2e^T x - x^T (H_C + I)\, x : 0 \leq x \leq e\}. \tag{3}$$

Similarly to the case of $\upsilon(C)$, it can be easily seen that $\alpha(G) \leq \upsilon^-(C)$ for all weighted adjacency matrices C. Hence from the $\upsilon^-(C)$ definition, we can assert that $\upsilon^-(C) \leq \upsilon(C)$ and, therefore,

$$\alpha(G) \leq \vartheta^-(G) \leq \vartheta(G), \tag{4}$$

i.e., $\vartheta^-(G)$ is an upper bound on $\alpha(G)$ that is not worse than $\vartheta(G)$.

This paper addresses the following question: in view of (4), can we use characterization (2) to approximate the maximum stable set of large graphs? To this end, an algorithm that runs in two phases is proposed. In the first phase, considering $\upsilon^-(C)$ as a function of the nonzero entries of the matrix C, a descent method is applied to decrease $\upsilon^-(C)$ as much as possible; for the weighted adjacency matrix C for which the best $\upsilon^-(C)$ was obtained, the second phase uses the corresponding optimal solution of $(P_G^-(C))$ to construct the input of a multi-start tabu search algorithm in order to obtain a near-optimal stable set of G.

There are several papers in the literature that try to use the Lovász theta number and/or its variants to extract stable sets. Grötschel, Lovász, and Schrijver [13] describe several alternative formulations of the Lovász theta number and provide a polynomial time algorithm for its computation based on the

ellipsoid method. In [13], they use this algorithm to determine the maximum stable set of a perfect graph. Alizadeh [1] proposes a randomized algorithm for computing the Lovász theta number and gives a procedure for the construction of a maximum clique in perfect graphs. Burer, Monteiro, and Zhang [6] study the semidefinite programming formulation of the theta number with additional low rank constraints and use nonlinear continuous programming techniques to extract near-optimal stable sets. Other authors use semidefinite programming techniques to compute the Lovász theta number and provide several strategies to extract near-optimal stable sets. We cite in this category Alon and Kahale [2], Benson and Ye [3], Gruber and Rendl [14], and Yildrim and Fan [22]. Our approach, on the other hand, is based on solving a sequence of convex quadratic programming problems to approximate the number ϑ^- defined in (2) and embedding the best obtained solution on a tabu search heuristic to reach a near-optimal stable set.

This paper is organized as follows. In Sec. 2, the proposed algorithm is described in detail. Specifically the number $\upsilon^-(C)$ is studied as a function of the nonzero entries of C and a descent strategy is given to allow the decreasing of this function. Then the multi-start tabu search heuristic is described and in Sec. 3, the results of applying this heuristic to some DIMACS benchmark graphs are reported and commented.

2. The Proposed Algorithm

Schematically, the proposed two-phase heuristic can be described as is shown in Fig. 1.

0. **Input:** A weighted adjacency matrix C of a graph G
   Output: A stable set of G
1. *PHASE I* : Use a *descent method* to approximately minimize $\upsilon^-(C)$.
2. *PHASE II* : Let x^* be the optimal solution associated with the best $\upsilon^-(C)$ obtained. Sort x^* and determine the correspondent ranking of the vertices of G. Use this ranking as the input of a *multi-start tabu search heuristic* to approximate the maximum clique of $\overline{G}$.

Fig. 1. The proposed two-phase heuristic.

Note that the initial matrix C can simply be the adjacency matrix of G. Also, for $0 \leq x^* \leq e$, x^* can be considered as an approximation of the characteristic vector of a maximum stable set of G. The sorting of x^* is usually done in descending order. This allows a ranking of the vertices of G where the first ones can be considered as the most serious candidates to be included in a maximum stable set of G. As we will see below, this ranking is explored by a multi-start procedure that generates a set of inputs for a tabu search routine.

It is obvious that, when applied to the complement of a graph G, this heuristic allows one to approximate the maximum clique of G. It is precisely what we have done in Sec. 3, where the heuristic is applied to the complements of 34 DIMACS clique benchmark graphs.

We now describe both phases in detail.

2.1. The descent method to decrease $\upsilon^-(C)$. Let C be a weighted adjacency matrix of a graph $G = (V, E)$. To implement phase I, we need to study $\upsilon^-(C)$ as a function of the nonzero entries of C (or, equivalently, as a function of the edges of G). If $|E| = m$, the family of weighted adjacency matrices associated with G can be considered as the image set of the following mapping:

$$\begin{aligned} C : \mathbb{R}^m \setminus \{0\} &\longrightarrow \mathbb{R}^{n \times n} \\ y &\longrightarrow C(y), \end{aligned}$$

where y is indexed by the edges of G. Hence the entries (i, j) and (j, i) of $C(y)$ are equal to y_{ij} if $ij \in E$. On the other hand, the entries of $C(y)$ corresponding to pairs (i, j) such that $ij \notin E$ are null. Thus, if y is a vector of ones, $C(y)$ is the adjacency matrix of G. Note also that $C(y)$ is a continuously differentiable function.

Since we are interested in studying $\upsilon^-(C)$ as a function of C nonzero entries, we can represent it by $\upsilon^-(C(y))$ or more simply by $\upsilon^-(y)$, where $y \in \mathbb{R}^m \setminus \{0\}$. Hence we can write

$$\upsilon^-(y) = \max_{0 \le x \le e} \upsilon_x^-(y) \tag{5}$$

with

$$\upsilon_x^-(y) = 2e^T x - x^T \left(\frac{C(y)}{-\lambda_{\min}(y)} + I \right) x, \tag{6}$$

where $-\lambda_{\min}(y) = -\lambda_{\min}(C(y))$.

Our aim is to decrease $\upsilon^-(y)$ using a descent method. Function $\upsilon^-(y)$ is, in general, nonconvex and nonsmooth on $\mathbb{R}^m \setminus \{0\}$. However, we can identify some elements of the generalized subdifferential $\partial\upsilon^-(y)$ (see [7]) that will allow us to construct a search direction to decrease $\upsilon^-(y)$.

As a matter of fact, the functions $\upsilon_x^-(y)$ are subdifferentiable for each x and, since $\upsilon^-(y)$ is finite, if $g \in \partial\upsilon_{x^*}^-(y)$ for some optimal solution x^* of (5), it can easily be concluded that $g \in \partial\upsilon^-(y)$. Hence using the subdifferential addition and quocient rules, we have

$$\partial\upsilon_{x^*}^-(y) = \partial \frac{x^{*T} C(y) x^*}{\lambda_{\min}(y)} = \frac{\lambda_{\min}(y)\ \partial\left(x^{*T} C(y) x^*\right) - \left(x^{*T} C(y) x^*\right)\ \partial\lambda_{\min}(y)}{[\lambda_{\min}(y)]^2}.$$

Note that $x^{*T} C(y) x^* = 2 \sum_{ij \in E} y_{ij} x_i^* x_j^*$ is differentiable in y and, therefore,

$$\partial\left(x^{*T} C(y) x^*\right) = \{\nabla x^{*T} C(y) x^*\},$$

where $\nabla x^{*T} C(y) x^*$ is the gradient vector with components $2x_i^* x_j^*$ for each $ij \in E$. On the other hand, it is well known that

$$\partial\lambda_{\min}(y) = \text{conv}\{\nabla u^T C(y) u : u \text{ is a normalized eigenvector associated with } \lambda_{\min}(y)\},$$

where "conv" denotes the convex hull and, similarly to the above, $\nabla u^T C(y) u$ has components $2u_i u_j$ for each $ij \in E$ (u_i and u_j are entries of a normalized eigenvector u associated with $\lambda_{\min}(y)$). Thus, we obtain a vector $g \in \partial\upsilon_{x^*}^-(y)$ if we define, for each entry y_{ij} of y, the corresponding component g_{ij} of g as follows:

$$g_{ij} = \frac{2}{\lambda_{\min}(y)} \left[x_i^* x_j^* + \frac{x^{*T} C(y) x^*}{-\lambda_{\min}(y)} u_i u_j \right]. \tag{7}$$

The vectors g of this form belong to $\partial\upsilon^-(y)$ and, consequently, it makes sense to use them to construct search directions to decrease $\upsilon^-(y)$. This construction is performed by Algorithm 1 described below, which corresponds to phase I of the proposed heuristic in Fig. 1.

Algorithm 1 uses ideas of the "gradient sampling algorithm" of Burke, Lewis, and Overton [4, 5] as well as the "aggregation of subgradients," which is typical of bundle algorithms for optimizing nonsmooth convex functions (see [16]).

Since $\upsilon^-(y)$ is finite and $\upsilon_x^-(y)$ is Lipschitz on V_y for each x such that $0 \le x \le e$, $\upsilon^-(y)$ is Lipschitz on some open neighborhood $V_y \subset \mathbb{R}^m \setminus \{0\}$, for any $y \in \mathbb{R}^m \setminus \{0\}$ (see [7]). Taking into account Rademacher's theorem, it is implicitly assumed in Algorithm 1 that $\upsilon^-(y)$ is differentiable almost everywhere in $\mathbb{R}^m \setminus \{0\}$.

Note also that we can restrict the y components to be between -1 and 1. In fact, let $C(y)$ be a weighted adjacency matrix for which the minimum in (2) is attained. If the greatest absolute value of the y components is K, we conclude that

$$H_{C(y)} = \frac{C(y)}{-\lambda_{\min}(y)} = \frac{KC(y/K)}{-K\lambda_{\min}(y/K)} = \frac{C(y/K)}{-\lambda_{\min}(y/K)} = H_{C(y/K)}$$

and, consequently, $\vartheta^-(G) = \upsilon^-(y) = \upsilon^-(y/K)$. Hence for minimizing $\upsilon^-(y)$ we can assume without loss of generality that all the y components belong to the interval $[-1, 1]$. Thus, after each iteration of Algorithm 1, the current iterate y is divided by its greatest absolute value entry.

Algorithm 1-A descent method to decrease $\upsilon^-(y)$

(1) (*Initialization*) Given a graph $G = (V, E)$ and its adjacency matrix C, choose the sampling ratio ε, the initial point $y = (y_{ij})_{ij \in E}$ that is equal to the vector of ones plus a vector whose entries are obtained by sampling from a uniform distribution on $[-\varepsilon/2, \varepsilon/2]$, a positive integer N defining the number of sampling points, an aggregation parameter β, and a positive integer M specifying the maximum number of iterations to be allowed. Set $k := 1$.

(2) (*Main iteration*) Carry out iteration k as follows:

(a) Form the set F in the following way:

(i) A set of vectors $\{g_z\} \subset \partial\upsilon^-(z)$ with entries given by (7), where z takes on $N+1$ values: the current iterate y, and N other vectors differing from y by vectors whose entries are obtained by sampling from a uniform distribution on $[-\varepsilon/2, \varepsilon/2]$;

(ii) A vector g_a resulting from the aggregation of the search directions computed before (if $k = 1$, F is only defined as in (i)).

(b) Define the search direction

$$d := -\arg\min\{\|s\|_2 : s \in \operatorname{conv}(F)\}.$$

If $d = 0$, go to Step 3.

(c) Use a line search to find a steplength t such that $\upsilon^-(y + td) < \upsilon^-(y)$. If $t > 0$, replace y by $y + td$, otherwise y is maintained (null step).

(d) If y was updated in 2(c), substitute y/K for y, where K is the maximum absolute value of the y components.

(e) Compute $g_y \in \partial\upsilon^-(y)$ with components given by (7) and determine the new search directions aggregation $\bar{g}_a$ by

$$\bar{g}_a := \begin{cases} (1-\beta)g_a + \beta g_y & \text{if} \quad t > 0, \\ g_a + g_y & \text{otherwise.} \end{cases}$$

Set $g_a := \bar{g}_a$.

(3) If $k > M$ or $d = 0$, compute $x^* = \arg\min\limits_{0 \le x \le e} \upsilon_x^-(y)$ and terminate; otherwise, increment k and return to step 2.

To decrease the objective function, we use a very simple line search that is similar to the one presented for the "gradient bundle algorithm" in [4].

Algorithm 1 was implemented in MATLAB. In Sec. 3, we will report on the computational results provided by its running on several DIMACS clique benchmark graphs.

2.2. The multi-start tabu search heuristic. Let x^* be the optimal solution provided by phase I of the proposed heuristic when applied to a graph G. Phase II is undertaken in order to approximate the maximum stable set of G by computing a high cardinality clique of $\overline{G}$. It begins by sorting x^* in descending order and by determining the sequence of nodes corresponding to the sorted x^*. Then a multi-start procedure will use this sequence to systematically generate subsequences from it. These generated subsequences represent the subgraphs of $\overline{G}$ that will constitute the initial solutions of the tabu search algorithm to obtain a high cardinality clique of $\overline{G}$.

One of most important tabu search features (see [10]) is based on the intensification and diversification strategies. Intensification performs the search in the most attractive regions that include the historically good solutions found. On the other hand, the diversification uses new regions to explore the unknown solutions space. The multi-start procedure is based on the same intensification-diversification idea, and it runs in two stages, as illustrated in Fig. 2.

The first stage, or intensification stage, starts from a subsequence constituted by the first node of the original sequence; then it adds to this subsequence the node that comes next in the original sequence thus forming a new subsequence; repeating this operation, successive subsequences are constructed until reaching the whole original sequence. Hence, from one subsequence to the next, the changes are

minimal and usually correspond to only one node, as described in Reingold et al. [21]. As an example, in Fig. 2 the first stage of the multi-start procedure begins with the subsequence 5 until reaching the original sequence 5, 4, 6, 2, 1, 3. At this stage the proximity to the good solutions, provided by the phase I, is explored.

After performing the combinations of the historically good regions, the search may continue in new regions. The second stage, or diversification stage, of the multi-start procedure consists of transforming each subsequence generated in the previous stage into a new subsequence by eliminating the furthest node on the left; repeating this operation, successive subsequences are obtained until reaching a subsequence with one node. Returning to Fig. 2, the subsequence 5, 4, 6, 2, 1 obtained at the first stage gives rise to subsequence 4, 6, 2, 1 and, eliminating successively the furthest node on the left, the subsequence formed by node 1 is reached.

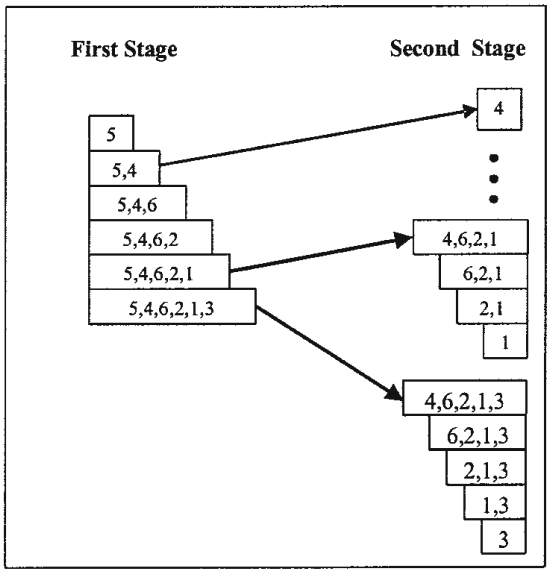

Fig. 2. Generation of subsequences of nodes in the multi-start procedure.

Each of the generated subsequences by the multi-start procedure constitutes the input of a tabu search routine (see [10]) to be described next. This improvement method combines two kinds of neighborhood structures: given a solution that is typically unfeasible, the method first undertakes to recover a feasible one; afterwards it attempts to increase the objective function value, as is done in Cavique, Rego, and Themido [8]. In order to implement these two operations, we consider a combined strategy (called Oscillation Strategy) which allows the solutions to come from admissible regions to nonadmissible ones and vice versa using the feasible and infeasible neighborhoods of Johnson [15].

To describe the mentioned neighborhood structures, some notation is needed (see Fig. 3). Define $A(S)$ as the set of vertices of G that are adjacent to vertices of a clique S. Let $k = |S|$ be the cardinality of S and $A_i(S)$ be the subset of vertices adjacent to i vertices in S. $A(S)$ can be divided into the subgroups $A_i(S)$ such that $A(S) = \bigcup_{i=1}^{k} A_i(S)$. The cardinality of the vertex set $V \setminus S$ is the sum of the number of vertices in $A(S)$ and the number of vertices nonadjacent to any vertex of S, whose set is denoted by $A_0(S)$. That is, $|V \setminus S| = \sum_{i=0}^{k} |A_i(S)|$.

The four neighborhood structures used by the algorithm are the following ones considered in [8]:

$$N^{+}_{\text{feas}}(S) = \{S' : S' = S \cup \{v\},\ v \in A_k(S),\ S \text{ and } S' \text{ are cliques}\},$$

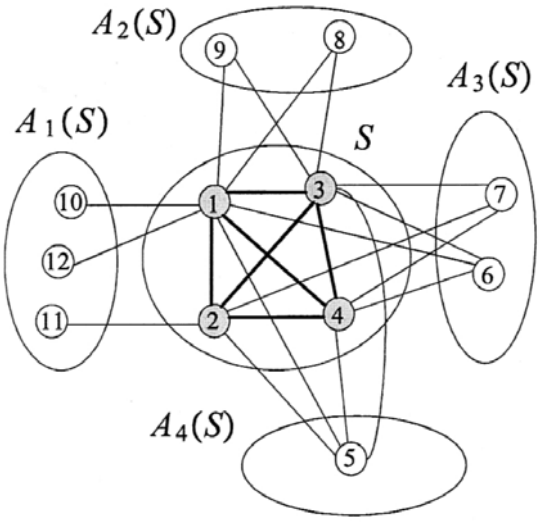

Fig. 3. Clique S and adajcent vertex sets $A_i(S)$.

$$N^0_{\text{feas}}(S) = \{S' : S' = S \cup \{v\} \backslash \{w\}, v \in A_{k-1}(S),\ w \in S,\ S \text{ and } S' \text{ are cliques}\},$$
$$N^+_{\text{infeas}}(S) = \{S' : S' = S \cup \{v\},\ v \in A_{k-1}(S),\ S \text{ might be a clique},\ S' \text{ is not a clique}\},$$
$$N^-_{\text{infeas}}(S) = \{S' : S' = S \backslash \{v\},\ v \in S,\ S \text{ is not a clique},\ S' \text{ might be a clique}\}.$$

Note that the subsets $A_k(S)$ and $A_{k-1}(S)$ are the most relevant for the algorithm. If $A_k(S) \neq \emptyset$, then it is possible to add one vertex to the current solution, thus increasing the clique value. If $A_{k-1}(S) \neq \emptyset$, then a vertex swap can be performed, maintaining the objective function value constant. For any other subset $A_{k-t}(S) \neq \emptyset$ it is possible to swap t solution vertices with one vertex of $A_{k-t}(S)$.

The tabu search routine is summarized below. We use the notation S for the current solution, S^* is the highest cardinality clique found so far, and T_1 and T_2 are tabu lists.

Tabu search routine

(1) (*Initialization*) Given a set of vertices S provided by the multi-start procedure which induces a subgraph of the complement graph $\overline{G}$, choose a positive integer M to terminate the algorithm, set $S^* = T_1 = T_2 = \emptyset$ and $k = 1$.

(2) (*Main iteration*) Carry out iteration k as follows:
   (a) if S is not a clique, go to step 2(e);
   (b) if there exists $S' \in N^+_{\text{feas}}(S)$ such that $(S' \backslash S) \backslash T_1 \neq \emptyset$, choose the best S' and go to step 2(f);
   (c) if there exists $S' \in N^0_{\text{feas}}(S)$ such that $(S' \setminus S) \setminus T_1 \neq \emptyset$, choose the best S', update T_1 and go to step 2(f);
   (d) choose the best S' in $N^+_{\text{infeas}}(S)$, update T_2 and go to step 2(f);
   (e) if there exists $S' \in N^-_{\text{infeas}}(S)$ such that $(S' \setminus S) \setminus T_2$, choose the best S';
   (f) set $S = S'$; if $|S| > |S^*|$ and S is a clique, set $S^* = S$.

(3) If $k > M$, terminate (S^* is the best obtained clique of $\overline{G}$); otherwise, increment k and return to Step 2.

Finally, note that there are similarities between the above-described multi-start tabu search procedure and the scatter search methods proposed in [9, 11, 12]. In both, the aim is to search diverse regions in the solution space using systematic methods, bypassing the random ones, so as to avoid time loss and repetition.

3. Computational Results

We begin by reporting on the results of applying Algorithm 1 of Sec. 2.1 to the complements of 34 DIMACS clique benchmark graphs. In this way, we obtained an approximation of the characteristic vector of the maximum clique for each of these graphs. The tests were carried out on a Pentium 4/1.6GHz with 512 MB RAM running under Windows XP. The interactive matrix language MATLAB (version 7.0) was used as well as the routine `quadprog.m` provided by its Optimization Toolbox for optimizing quadratic programming problems.

Algorithm 1 performed no more than two hours on each graph. For all graphs the following parameters were considered: $\varepsilon = 0.1$ (sampling ratio), $N = 20$ (number of sampling points), and $\beta = 0.7$ (aggregation parameter), as well as a sufficiently large number of iterations (namely $M = 500$) in order to allow the algorithm to run for at least two hours. Additionally, a stop instruction at the end of each iteration is used if υ^- does not decrease more than 0.001 in the last 10 iterations.

In the first seven columns of Table 1, we collected the following information and results: the graph's name, its number of nodes (n), its clique number (ω), the best approximation of the Lovász theta number known to us (ϑ), and the υ^- values obtained, respectively, at the beginning, after 1/2 hour running the algorithm's, and at the end of the algorithm (the latter one is naturally the best obtained υ^- value). Another column was added to Table 1 to record the running time (in hours). Note that, for several graphs the running time is less than two hours, a fact that resulted from an insufficient υ^- decrease in the last 10 iterations.

From the results presented in Table 1 some conclusions are in order. For all graphs, the values υ^- recorded after the first 1/2 hour and at termination represent a dramatic decrease of the initial value υ^-. Also, for the graphs whose running time was greater than 1/2 hour, the main decrease of υ^- took place during the first 1/2 hour running period. In fact, for all graphs, the decrease in the last iterations is much smaller than in the first ones. This is illustrated in Fig. 4 for the complement of graph `c-fat500-10`, where the highest difference between the "1/2 hour" and the "best obtained" υ^- values were observed. This figure shows a plot of the value υ^- as a function of the iteration number. The vertical line at iteration number 18 marks the 1/2 hour.

On the other hand, for the 25 graphs for which we know the exact theta number or its approximation, we computed the relative error of the value υ^-. The following positive remarks can be made: (1) the relative errors mean is about 22% for the "1/2 hour" values and about 10% for the best obtained values; (2) the maximum relative error for the "1/2 hour" values was about 80% and for the best obtained value was about 50%; both of these marks were attained only in two cases, `hamming8-4` and `p_hat500-1`; (3) the clique number was reached in five cases, the relative error is less than 15% in 15 of the "1/2 hour" values and less than 10% in 22 of the best obtained values.

These results show that the use of a descent method to decrease υ^- (as, for example, the method implemented by Algorithm 1) yields an approximation of ϑ^- in a reasonable time. Therefore, we can assert that the characterization (2) is a valuable tool for approximating ϑ^- (and naturally for the Lovász theta number ϑ). Obviously this leads to the idea that trying to improve Algorithm 1 or devising new methods to decrease υ^- is a worthwhile task in the future.

We now report on the application of the heuristic proposed in Fig. 1 to the complements of the same 34 DIMACS clique benchmark graphs considered above. This allows us to obtain for each of these graphs an approximation of its maximum clique. The reported application of Algorithm 1 to the complements of those benchmark graphs constitutes phase I of the heuristic. The outputs obtained in this phase are used in phase II to approximate the maximum clique of each graph.

The tests performed for phase II were carried out on a Pentium 4/2.8GHz with 256 MB RAM running under Windows XP. The computer program was written in C language and the Microsoft Visual C++ compiler was used.

The results are summarized in Table 2, where for each instance the following values were recorded: the instance's name, the number of nodes (n), the clique number (ω), the cardinality of the best

				υ^- values			
Graph	n	ω	ϑ	initial	1/2 hour	best value	run time
brock200_1	200	21	≤ 27.48	40.99	28.99	28.98	1:02h
brock200_2	200	12	≤ 14.37	25.31	15.78	15.60	2:00h
brock200_3	200	15	≤ 18.88	31.58	20.36	20.25	1:30h
brock200_4	200	17	≤ 21.32	34.65	22.80	22.57	1:30h
brock400_1	400	27	≤ 39.89	61.47	44.66	41.61	2:00h
brock400_2	400	29	≤ 39.73	61.93	45.66	41.89	2:00h
brock400_3	400	31	≤ 39.65	61.92	44.94	41.86	2:00h
brock400_4	400	33	≤ 39.77	62.07	45.84	41.70	2:00h
c-fat200-1	200	12	12	17.39	13.83	13.66	2:00h
c-fat200-2	200	24	24	32.73	25.65	25.32	2:00h
c-fat200-5	200	58	58	72.59	–	60.35	0:17h
c-fat500-1	500	14	–	20.55	18.56	16.44	2:00h
c-fat500-10	500	126	126	163.4	143.58	129.25	2:00h
hamming6-4	64	4	≤ 4.34	13.54	5.34	5.34	0:23h
hamming8-4	256	16	16	45.25	29.64	22.97	2:00h
keller4	171	11	≤ 13.94	41.15	15.15	15.05	1:47h
MANN_a9	45	16	≤ 17.19	19.71	17.49	17.49	0:04h
MANN_a27	378	126	≤ 132.6	230.9	133.12	133.05	2:00h
p_hat300-1	300	8	≤ 10.76	22.14	18.27	11.75	2:00h
p_hat300-2	300	25	–	52.73	30.25	29.20	2:00h
p_hat500-1	500	9	≤ 15.39	30.06	28.02	23.80	2:00h
san200_0.7_1	200	30	30	93.74	–	30	0:07h
san200_0.7_2	200	18	–	108.5	19.65	19.43	1:45h
san200_0.9_1	200	70	70	113.5	–	70	0:03h
san200_0.9_2	200	60	60	95.93	–	60	0:03h
san200_0.9_3	200	44	–	85.95	46.33	46.32	0:64h
san400_0.5_1	400	13	13	176.8	21.04	18.57	2:00h
san400_0.7_1	400	40	40	184.8	40.02	40	0:59h
san400_0.7_2	400	30	–	181.3	33.51	31.22	2:00h
san400_0.7_3	400	22	–	180.5	30.01	26.75	2:00h
san400_0.9_1	400	100	100	188.9	–	100	0:11h
sanr200_0.7	200	18	–	36.96	24.95	24.91	1:16h
sanr200_0.9	200	42	–	64.75	49.98	49.98	0:31h
sanr400_0.7	400	20	–	55.66	43.91	36.56	2:00h

Table 1. Algorithm 1 computational results.

solution found, and the time until the best solution in seconds, for three run sets: using a random start sequence and a time limit of 120 seconds, using a random start sequence without the time limit, and finally, the informed start column shows the results of the multi-start tabu search procedure using the data provided by phase I as input.

The main purpose of this work is to use the information provided by the number υ^- defined in (5) to find an approximation of the maximum stable set (or the maximum clique) of challenging instances. We define a challenging instance as an instance that cannot be solved by the multi-start tabu search procedure with random start in less than 120 seconds.

Taking into account the tabu search efficiency, only the `brock` and `san` instances as well as the `MANN_a27` graph can be considered challenging in our experiments. Therefore, in Table 2, we only show the results relative to the random start without the 120 seconds limit and to the informed start for these challenging instances.

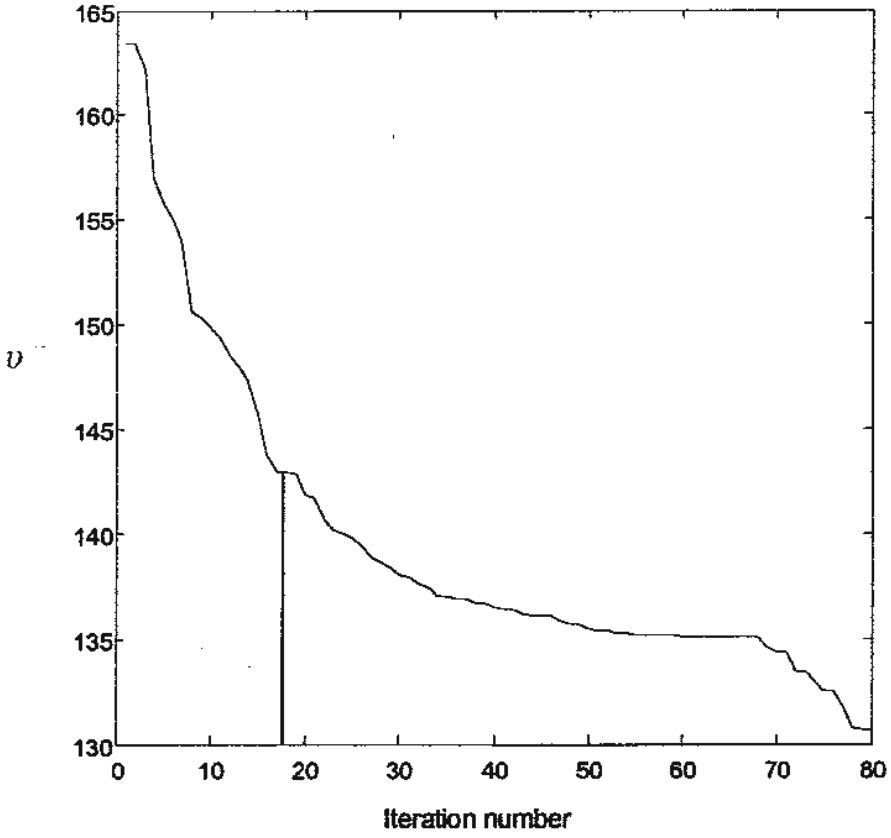

Fig. 4. The v^- value decreasing for the complement of graph `c-fat500-10`.

The `brock` and `san` instances were specially conceived to trick the local search procedures. However, we can see that the approach allows us to obtain the maximum clique for all the challenging instances with the exception of the `brock400_1` and `brock400_2`. Note also that, using the data provided in phase I, the optimal solution can be reached for the `brock200_3`, while in the random version the optimum is not reached.

Concerning the computational times, comparing columns 2 and 3, significant reduction is obtained in the `brock200_3`, `san200_0.7_2`, `san400_0.7_1`, `san400_0.7_2` and `san400_0.9_1` instances. These results confirm that informed starts can be very useful in the multi-start tabu search procedure, a fact that is recognized by many researchers. Actually, the combination of heuristics like those of phases I and II seems to be a very promising research area and will be part of our future work.

Acknowledgments. The authors acknowledge the referee for the valuable comments and suggestions which improved the paper. The second author's research was supported by the *Centre for Research on Optimization and Control* (CEOC) from the *Fundação para a Ciência e a Tecnologia* (FCT), cofinanced by the European Community Fund FEDER/POCI 2010.

Graph	n	ω	1. Random start Time < 120 sec.		2. Random start Time > 120 sec.		3. Informed start provided by phase I	
			Card. best solution	Time (secs.)	Card. best solution	Time (secs.)	Card. best solution	Time (secs.)
brock200_1	200	21	21	8	–	–	–	–
brock200_2	200	12	11	1	12	541	12	13
brock200_3	200	15	14	1	14	1352*	15	16
brock200_4	200	17	16	1	17	177	17	75
brock400_1	400	27	24	5	25	35291*	25	35655*
brock400_2	400	29	24	22	25	35147*	25	35238*
brock400_3	400	31	24	7	31	5212	31	9237
brock400_4	400	33	24	4	33	646	33	539
c-fat200-1	200	12	12	1	–	–	–	–
c-fat200-2	200	24	24	1	–	–	–	–
c-fat200-5	200	58	58	2	–	–	–	–
c-fat500-1	500	14	14	3	–	–	–	–
c-fat500-10	500	126	126	52	–	–	–	–
hamming6-4	64	4	4	<1	–	–	–	–
hamming8-4	256	16	16	<1	–	–	–	–
keller4	171	11	11	1	–	–	–	–
MANN_a9	45	16	16	<1	–	–	–	–
MANN_a27	378	126	125	24	126	5043	126	1080
p_hat300-1	300	8	8	<1	–	–	–	–
p_hat300-2	300	25	25	4	–	–	–	–
p_hat500-1	500	9	9	4	–	–	–	–
san200_0.7_1	200	30	30	1	–	–	–	–
san200_0.7_2	200	18	17	103	18	403	18	4
san200_0.9_1	200	70	70	1	–	–	–	–
san200_0.9_2	200	60	60	6	–	–	–	–
san200_0.9_3	200	44	44	23	–	–	–	–
san400_0.5_1	400	13	12	16	13	1267	13	1224
san400_0.7_1	400	40	22	11	40	1403	40	4
san400_0.7_2	400	30	18	2	30	1368	30	10
san400_0.7_3	400	22	18	58	22	1118	22	1250
san400_0.9_1	400	100	55	12	100	1868	100	9
sanr200_0.7	200	18	18	4	–	–	–	–
sanr200_0.9	200	42	42	<1	–	–	–	–
sanr400_0.7	400	20	20	2		–	–	–

*total time for non-optimal solutions

Table 2. Phase II computational results

REFERENCES

1. F. Alizadeh, "A sublinear-time randomized parallel algorithm for the maximum clique problem in perfect graphs," In: *ACM-SIAM Symposium on Discrete Algorithms*, Vol. 2, 188–194 (1991).
2. N. Alon and N. Kahale, "Approximating the independence number via the theta-function," *Math. Program.*, **80**, No. 3, 253–264 (1988).
3. S. Benson and Y. Ye, "Approximating maximum stable set and minimum graph coloring problems with the positive semidefinite relaxation," In: *Applications and Algorithms of complementarity, Eds. M. Ferris and J. Pang* , Kluwer Academic Publishers, (2000), pp. 1–18.
4. J. V. Burke, A. S. Lewis, and M. L. Overton, "Two numerical methods for optimizing matrix stability," *Linear Algebra Appl.*, **351–352**, 117–145 (2002).

5. J. V. Burke, A. S. Lewis, and M. L. Overton, "A robust gradient sampling algorithm for nonsmooth, nonconvex optimization," *SIAM J. Optim.*, **15**, No. 3, 751–779 (2005).
6. S. Burer, R. D. C. Monteiro, and Y. Zhang, "Maximum stable formulations and heuristics based on continuous optimization," *Math Program.*, **94**, No. 1, 137–166 (2002).
7. F. H. Clarke, "Optimization and nonsmooth analysis," *Classics Appl. Math.*, **5**, SIAM, Philadelphia (1990).
8. L. Cavique, C. Rego, and I. Themido, "Estruturas de vizinhança e procura local no problema da clique máxima," *Investigacón Oper.*, **22**, 1–18 (2002).
9. L. Cavique, C. Rego, and I. Themido, "A scatter search algorithm for the maximum clique problem," In: *Essays and Surveys in Metaheuristics, eds. C. Ribeiro and P. Hansen*, Kluwer Academic Publishers (2002), pp. 227–244.
10. F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic Publishers (1997).
11. F. Glover, "A template for scatter search and path relinking," In: *Lecture Notes in Computer Science, Eds. J.-K. Hao, E. Lutton, E. Ronald, M. Schoenaurer and D. Snyers* (1997), pp. 1–5.
12. F. Glover, "Scatter search and path relinking," In: *New Ideas in Optimization* (D. Corne, M. Dorigo, and F. Glover, eds.), McGraw-Hill International (1999).
13. M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer, Berlin (1988).
14. G. Gruber and F. Rendl, "Computational experience with stable set relaxations," *SIAM J. Optim.*, **13**, No. 4, 1014–1028 (2003).
15. D. S. Johnson, "Approximation algorithms for combinatorial problems," *J. Comput. System Sci.*, **9**, 256–278 (1974).
16. K. C. Kiwiel, "Methods of descent for nondifferentiable optimization," *Lecture Notes in Math.*, **1133**, Springer-Verlag, Berlin, New York (1985).
17. D. E. Knuth, "The sandwich theorem," *Electron. J. Combin.*, **1**, Article 1 (1994).
18. L. Lovász, "On the Shannon capacity of a graph," *IEEE Trans. Inform. Theory*, **25**, No. 2, 1–7 (1979).
19. C. J. Luz, "An upper bound on the independence number of a graph computable in polynomial time," *Oper. Res. Lett.*, **18**, 139–145 (1995).
20. C. J. Luz and A. Schrijver, "A convex quadratic characterization of the Lovász theta number," *SIAM J. Discrete Math.*, **19**, No. 2, 382–387 (2005).
21. E. M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms: Theory and Practice*, Prentice-Hall (1977).
22. E. A. Yildirim and X. Fan, "On extracting maximum stable sets in perfect graphs using Lovász's theta function," *Comput. Optim. Appl.*, **33**, Nos. 2-3, 229–247 (2006).

L. Cavique
Escola Superior de Comunicação Social, Instituto Politécnico de Lisboa, Lisboa, Portugal
E-mail: lcavique@escs.ipl.pt
C. J. Luz
Escola Superior de Tecnologia de Setúbal, Instituto Politécnico de Setúbal, Setúbal, Portugal
E-mail: cluz@est.ips.pt