GRASP and Path Relinking for the Matrix Bandwidth Minimization*

Estefanía Piñana, Isaac Plana, Vicente Campos and Rafael Martí

Dpto. de Estadística e Investigación Operativa,
Facultad de Matemáticas, Universitat de Valencia
Dr. Moliner, 50, 46100 Burjassot-Valencia, Spain.

Latest version: November 12, 2001

Abstract — In this article we develop a greedy randomized adaptive search procedure (GRASP) for the problem of reducing the bandwidth of a matrix. This problem consists of finding a permutation of the rows and columns of a given matrix, which keeps the nonzero elements in a band that is as close as possible to the main diagonal. The proposed method may be coupled with a Path Relinking strategy to search for improved outcomes. Empirical results indicate that the proposed GRASP implementation compares favourably to classical heuristics. GRASP with Path Relinking is also found to be competitive with a recently published tabu search algorithm that is considered one of the best currently available for bandwidth minimization.

Keywords: Metaheuristics, GRASP, Path Relinking, Matrix Bandwidth

* Research partially supported by the *Ministerio de Ciencia y Tecnología* of Spain with code TIC2000-1750-C06-01.

1. Introduction

The matrix bandwidth minimization problem (MBMP) has been the subject of study for at least 32 years, beginning with the Cuthill - McKee algorithm in 1969. The problem consists of finding a permutation of the rows and the columns of a matrix that keeps all the non-zero elements in a band that is as close as possible to the main diagonal. This problem has generated considerable interest over the years because of its practical relevance for a significant range of global optimization applications. They include preprocessing the coefficient matrix for solving the system of equations, finite element methods for approximating solutions of partial differential equations, large-scale power transmission systems, circuit design, hypertext layout, chemical kinetics and numerical geophysics.

Given a matrix $A=\{a_{ij}\}_{nxn}$ the problem can be stated in terms of graphs considering a vertex for each row (column) and an edge in E as long as either $a_{ij} \neq 0$ or $a_{ji} \neq 0$. The problem consists of finding a labeling f of the vertices that minimizes the maximum difference between labels of adjacent vertices. In mathematical terms, given a graph $G=(V,E)$ with vertex set V ($|V|=n$) and edge set E, we seek to minimize:

$$B_f(G) = \max\{B_f(v) : v \in V\} \text{ where } B_f(v) = \max\{|f(v) - f(u)| : u \in N(v)\}.$$

In this expression, $N(v)$ is the set of vertices adjacent to v, $f(v)$ is the label of vertex v and $B_f(v)$ is the bandwidth of vertex v. A labeling f of G assigns the integers $\{1, 2, \ldots, n\}$ to the vertices of G; thus, it is simply a renumbering of these vertices. Then, the bandwidth of a graph is $B(G)$, the minimum $B_f(G)$ value over all possible labelings f. The MBMP consists of finding a labeling f that minimizes $B_f(G)$. Table 1 summarizes some of the relevant work in the area to the present.

Table 1 Summary of relevant literature.

Reference	Procedure	Comments
Cuthill and McKee (1969)	Reverse Method	First known method
Gibbs, Poole and Stockmeyer (1976)	GPS	Takes advantage of the graph structure
Luo (1992)	GPS variant	Special structured graphs
Dueck and Jeffs (1995)	Simulated Annealing	Long CPU time
Martí et al. (2001)	Tabu search	Includes a comparative study of heuristic approaches

The main application of this problem is to solve nonsingular systems of linear algebraic equations of the form $Ax = b$. The preprocessing of A to reduce its bandwidth results in substantial savings in the computational effort associated with solving the system of equations. For many years researchers were only interested in designing relatively simple heuristic procedures and sacrificed solution quality for speed. This is the case of the Reverse Method and the GPS procedure. These two methods yield similar results in terms of solution quality, however GPS is considerably faster, with an average speed that is about 8 times faster than the reverse Cuthill-McKee procedure.

Luo (1992) proposes an algorithm to reduce both, the bandwidth and the profile of a sparse matrix. The method is based on some refinements of GPS and it is more complex. Computational results are given over a set of special structured graphs; these results however, are difficult to extend to general graphs.

Recently, metaheuristic methods have been applied to the MBMP. A Simulated Annealing (SA) procedure was first introduced by Dueck and Jeffs (1995). It is based on an insertion mechanism and does not take advantage of the graph structure as the

GPS method does. The experimentation shows that this SA implementation is inferior to GPS on "grid", "path", circle", "windmill" and "st" graphs. However, SA outperforms GPS in terms of solution quality on ternary trees, binary trees and random graphs. It should be noted that, although SA finds better labelings than GPS in some graphs, it does so by taking up to 2000 times longer.

Martí et al. (2001) propose a tabu search method for this problem. It is likewise based on swap moves that exchange the labels of a pair of vertices. The operator $move(v,u)$ assigns the label $f(u)$ to vertex v and the label $f(v)$ to vertex u. Since the objective is to change the labels in order to reduce the current value of $B_f(G)$, a *candidate list* of moves based on a set $C(f)$ of critical and near-critical vertices is constructed:

$$C(f) = \{v : B_f(v) \geq \alpha * B_f(G)\}$$

where $1 > \alpha > 0$. In order to construct a candidate list of moves based on the vertices in $C(f)$, a set of suitable swapping vertices for each vertex in $C(f)$ must be found. The following two quantities for a vertex v and a labeling f are introduced:

$$max(v) = \max\{f(u) : u \in N(v)\} \qquad min(v) = \min\{f(u) : u \in N(v)\}$$

A suitable label for v in the current labeling f is given by *mid(v)*, then the set of suitable swapping vertices for v is defined as *N'(v)* which considers all vertices u with labels $f(u)$ that are "closer" to $mid(v)$ than $f(v)$.

$$N'(v) = \{u : |mid(v) - f(u)| < |mid(v) - f(v)|\} \text{ where } mid(v) = \left\lfloor \frac{\max(v) + \min(v)}{2} \right\rfloor$$

Let $CL(v) = \{move(v,u) : u \in N'(v)\}$ be the candidate list of moves associated with a vertex $v \in C(f)$. The value of a $move(v,u)$ is defined as the number of vertices adjacent to v or u (including u) whose bandwidth increases due to the move. The authors propose a flexible implementation of this criterion, and if a vertex increases its bandwidth marginally with respect to the bandwidth of the graph, it is considered that this increment has no influence on the solution's value. Specifically, it is said that the bandwidth of vertex z has experienced more than just a marginal increase, and is considered in the move value computation, if $B_{f'}(z) > \beta B_f(G)$. After experimentation, α and β were set to 0.2 and 0.8, respectively.

The basic tabu search implementation consists of a short-term memory design in which the identity of a vertex whose label has been changed is the attribute used to impose a tabu restriction. Specifically, after a $move(v,u)$ is executed, the labels of vertices v and u are not allowed to change until the tabu tenure expires. A longer-term diversification based on frequency information is added to this basic implementation. The method incorporates a memory structure to record information about previously found solutions. This information is used to re-start the search with a new labeling f, which is built considering both diversity and quality criteria.

Martí et al. (2001) present a computational comparison of the GPS procedure, the SA method and two variants of the TS algorithm (with and without the restarting mechanism). The experiments are performed over 113 instances from the *Harwell-Boeing Sparse Matrix Collection,* which consists of a set of standard test matrices from a wide variety of scientific and engineering disciplines. It is shown that SA yields inferior solutions compared to TS and it takes much more running time than TS. The basic implementation of TS is superior to GPS in terms of solution quality and competitive in terms of speed in the small instances. However, in the large instances, TS cannot compete with GPS in terms of time. The TS version with restarting is

robust in terms of solution quality, with an average deviation from the best-known solutions of 5% for the longer runs. On the other hand, GPS is capable of generating good solutions at a speed that is hard to match by a metaheuristic. Nevertheless, tabu search finds increasingly high-quality solutions when the search is allowed to go beyond a few (approximately 5) CPU seconds. It should also be mentioned that GPS performs best on instances with structured graphs. That is, the more structure in the graph the better the performance of GPS.

Our paper presents the results of applying a *Greedy Randomized Adaptive Search Procedure* (GRASP, Feo and Resende 1995) to the MBMP. Each GRASP iteration consists of constructing a trial solution and then applying an improvement procedure, typically a local search method, to find a local optimum (i.e., the final solution for that iteration). Performing multiple GRASP iterations may be interpreted as a means of strategically sampling the solution space. We also present a search procedure that combines GRASP with *Path Relinking* as a form of intensification (Laguna and Martí, 1999) to search for improved outcomes. This approach generates new solutions by exploring trajectories that connect high-quality solutions previously found with the GRASP method.

This paper is organized as follows. The following section discusses the constructive method of our GRASP implementation. We have developed and tested five different constructive methods with the goal of selecting the best one that balances intensification and diversification in the search. Section 3 is devoted to the local search phase of the GRASP algorithm. Section 4 presents the path relinking method that finds a path between two "good" solutions previously generated by GRASP, in order to discover new potentially better ones. Finally, Section 5 presents extensive computational experimentation with the *Harwell-Boeing Sparse Matrix Collection*. This section also introduces an integer linear formulation of the MBMP that allows us to compute the optimum solution for limited size instances. Finally, we outline our conclusions and directions for future research.

2. GRASP Construction Phase

The GRASP construction phase is iterative, greedy, and adaptive. It is iterative because the initial solution is built considering one element at a time. It is greedy because the addition of each element is guided by a greedy function. It is adaptive because the element chosen at any iteration in a construction is a function of those previously chosen. That is, the method is adaptive in the sense of updating relevant information from one construction step to the next.

This phase of the GRASP methodology is particularly important, given the goal of developing a method that balances diversification and intensification in the search. For this purpose we have developed and tested five different constructive methods. We first describe constructive method C1:

This phase starts by creating a list of unlabeled vertices U, which at the beginning consists of all the vertices in the graph (i.e., initially U=V). The first vertex v is randomly selected from all those vertices in U. We assign a random label to v and delete it from U. In subsequent construction steps, the candidate list CL consists of all the vertices in U that are adjacent to at least one labeled vertex. The restricted candidate list RCL is formed from those vertices in CL with a maximum number of adjacent labeled vertices. A vertex v is randomly selected from RCL in order to be labeled.

Once v is selected it is labeled with the best available label according to the vertices already labeled. In mathematical terms, the best label for vertex v is *mid(v,U)*:

$$max(v,U) = \max\{ f(u) : u \in N(v) \cap (V - U) \},$$

$$min(v,U) = \min\{ f(u) : u \in N(v) \cap (V - U) \},$$

$$mid(v,U) = \left\lfloor \frac{\max(v,U) + \min(v,U)}{2} \right\rfloor.$$

Vertex v is labeled with the closest available label to *mid(v,U)* (i.e., the closest number not yet assigned to a previously selected vertex). Then U, CL and RCL are updated for the next step. The construction phase terminates after n steps, when all vertices have been selected and labeled (i.e., when $U = \varnothing$).

Constructive methods C2 and C3 are the same as C1 except for the definition of the RCL. The restricted candidate list RCL is formed, at each step of C2, with those vertices that have been in CL for a maximum number of construction steps. C3 combines both criteria by considering the attractiveness of a vertex u as the sum of both measures: the number of adjacent labeled vertices to u, and the number of steps that u has been in CL. Both measures are scaled in order to add them up, and the vertices with maximum attractiveness are included, at each step, in the RCL.

Constructive methods C4 and C5 are partially based on the construction of a level structure of V proposed in the GPS method. A level structure is a partition of V into sets $L_1, L_2, \ldots, L_k$, called levels, with the following characteristics:

- Vertices adjacent to a vertex in level L_1 are in either L_1 or L_2
- Vertices adjacent to a vertex in L_k are in either L_k or L_{k-1}
- Vertices adjacent to a vertex in L_i (for $1 < i < k$) are in either L_{i-1}, L_i or L_{i+1}

The constructive procedure C4 consists of the following two phases:

1. *Finding a level structure*: This procedure starts by randomly selecting a vertex (root) from those vertices of low degree. L_1 consists only of this root. Then, the algorithm constructs a level structure from this starting vertex. As in the GPS method, once all the vertices have been assigned to levels, a backward step constructs a second level structure, beginning with a vertex in the last level with minimum degree. The procedure now combines these two level structures into one new structure whose width is usually less than that of either of the original ones. This first phase can be considered a pre-processing of the GRASP construction.

2. *Labeling*: The labeling procedure assigns, level-by-level, consecutive labels to the vertices V of G beginning with label 1. At each iteration of the construction phase, the candidate list CL is formed with those vertices in the current level. Therefore, initially CL=L_1, and when all the vertices in L_1 have been labeled, CL=L_2 and so on, until all the vertices in L_k have been labeled

   Given a label l to be assigned (labels 1,2,.., l-1, have been previously assigned), and a vertex v in level L_i, we define *LeftB(v,l)* as the difference between l and the minimum label of its adjacent vertices in L_{i-1}. We also define *RightB(v,l)* as the difference between the maximum label of its adjacent vertices in L_{i+1} and l. If label l is assigned to vertex v, then $B_f(v)$ would be the maximum between *LeftB(v,l)* and *RightB(v,l)*. However, since the vertices in level L_{i+1} are not yet labeled, we cannot compute *RightB(v,l)* exactly. We compute a lower bound of this value as the number of unlabeled vertices in L_i (excluding v) plus the number of adjacent vertices of v in L_{i+1}. If we wait until the next step to label vertex v (with label $l+1$), *LeftB(v,l)* will increase in a unit, while *RightB(v,l)* will decrease in a unit. Therefore, if *LeftB(v,l)* is greater than *RightB(v,l)* it is better to label vertex v as soon as possible in order to obtain a low $B_f(v)$ value; on the other hand, if it is lower, there is no need to label it now (and thus we can wait until

later steps). Then, the restricted candidate list of vertices RCL is built with those vertices v in CL with $LeftB(v,l) > RightB(v,l)$. A vertex v is randomly selected from RCL and labeled with l. The construction phase terminates after n steps, when all vertices have been selected and labeled.

The example in Figure 1 illustrates these computations in the construction of the RCL.

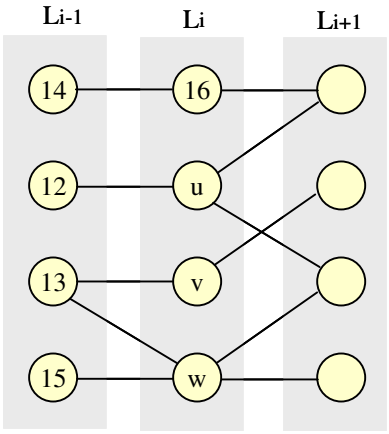

Figure 1. Restricted Candidate List

Figure 1 shows a partial representation of a graph where levels L_{i-1}, L_i and L_{i+1} are depicted. Consider that all the vertices in level L_{i-1} have been labeled (the number of each vertex represents its label). Therefore, at this iteration CL=*{u, v, w }* and we have to select a vertex to assign label 17. Following the above calculations:

$$LeftB(u,17) = 5 \qquad RightB(u,17) \geq 2 + 2 = 4$$
$$LeftB(v,17) = 4 \qquad RightB(v,17) \geq 2 + 1 = 3$$
$$LeftB(w,17) = 4 \qquad RightB(w,17) \geq 2 + 2 = 4$$

Then, RCL = *{u,v}* and we randomly select a vertex from this set to assign label 17.

If vertex v does not have any adjacent vertex in level L_{i-1} we do not compute *LeftB(v,l)* as before, but we assign *LeftB(v,l)* a value of 0, forcing the method to not label v until later steps. Similarly, if vertex v does not have any adjacent vertex in level L_{i+1}, we compute *RightB(v,l)* as the number of unlabeled adjacent vertices to v in L_i. If no vertex in CL satisfies the condition $LeftB(v,l) > RightB(v,l)$, then RCL = CL.

Constructive method C5 is equal to C4 except for the criteria to build RCL from CL. Specifically, those vertices in CL with a minimum value of *RightB(v,l)-LeftB(v,l)* are included in RCL. In the example given in Figure 1, u and v also meet this criteria, thus RCL = *{u,v}* in this case.

2.1 Quality and Diversity

We now propose a measure of diversity and a measure of quality to compare the performance of the 5 competing constructive methods. A "good" GRASP constructive algorithm should provide solutions with a good objective function value as well as being sufficiently scattered in the solution space to allow the local search phase to reach different local optima. We have selected 10 representative instances of different sizes and densities from the *Harwell-Boeing Sparse Matrix Collection* to compare the 5 methods. We have generated 100 different solutions with each method on each instance.

A measure of the quality of a method is the average objective function from among the 100 generated solutions. A measure of the diversity of a method is the average distance of all the pairs of the 100 labelings (solutions) generated by the method. We define the distance between two labelings (permutations) $p = (p_1, p_2, ..., p_n)$ and $q = (q_1, q_2, ..., q_n)$ as:

$$d(p,q) = \sum_{i=1}^{n} |p_i - q_i| .$$

To compare quality and diversity, we have scaled both measures in a such a way that they belong to the interval [0,1]. Moreover, we have computed the quality measure as 1 minus the scaled value. In this way, the greater the quality and diversity values, the better the method. Figure 2 shows the average of both measures across the 10 instances considered, on each constructive algorithm. This figure also shows the results with a pure random generator and the GPS and Tabu solutions as well, as a baseline for comparison.

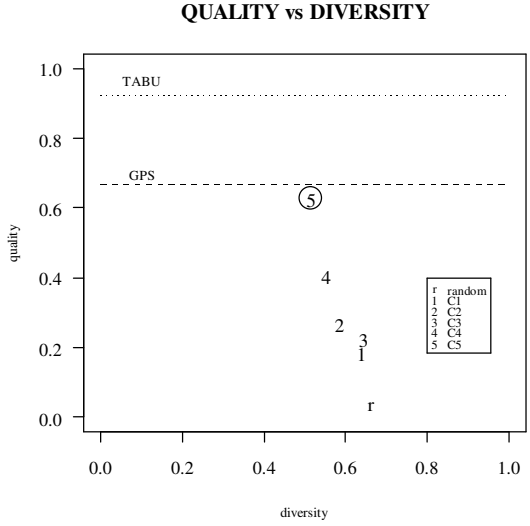

Figure 2. Constructive methods

As expected, the random generator produces the maximum diversity. C1 and C3 almost match the diversity of the random method using a systematic approach instead of randomness; however, they produce low quality solutions. None of these constructive methods produce on average solutions which are as good as the GPS or Tabu methods. We have selected C5 as the constructive method of our GRASP algorithm since it provides a good balance between diversity and quality.

3. GRASP Improvement Phase

The GRASP improvement phase typically consists of a local search procedure. Our local search method is partially based on the tabu search algorithm proposed in Martí et al. (2001). We consider the set of critical vertices $C(f) = \{v : B_f(v) = B_f(G)\}$. Note that we do not consider the near-critical vertices in this set (as the tabu search method does) since we have computationally found that their inclusion does not add any value to the local search algorithm. We have also considered the operator *move(u,v)*, and the candidate list of moves *CL(v)* associated with a vertex $v \in C(f)$ as described in Section 1. We propose a new move evaluation and study two different strategies for move selection.

The value of a $move(v,u)$ is the difference between the number of critical vertices before and after the move. In mathematical terms:

$$MoveValue(v,u) = |C(f)| - |C(f')|$$

where f' is the labeling obtained when applying *move(v,u)* to the current labeling f. A positive *MoveValue* indicates that the solution "improves" although the objective value may or may not be reduced.

Each step of the local search consists of selecting a vertex v in *C(f)* to be considered for a move. *CL(v)* is computed, and a vertex u is selected in *CL(v)* to perform *move(u,v)*. Two selection strategies were considered during preliminary experimentation. Given a vertex v, the *best* strategy selects the move *move(v,u)* with the largest move value from among all the moves with u in *CL(v)*. The *first* strategy, on the other hand, scans *CL(v)* in search of the first vertex u whose movement results in a strictly positive move value. Section 5 contains the computational comparison of both strategies. The local search phase terminates when improvement is no longer possible (i.e: when there is no move that reduces the number of vertices in C(f)).

4. Path Relinking

Path relinking has been suggested as an approach to integrate intensification and diversification strategies in the context of tabu search (Glover and Laguna, 1997). This approach generates new solutions by exploring trajectories that connect high-quality solutions, by starting from one of these solutions, called an *initiating solution*, and generating a path in the neighbourhood space that leads toward the other solutions, called *guiding solutions*. This is accomplished by selecting moves that introduce attributes contained in the guiding solutions. Path relinking in the context of GRASP, was first introduced by Laguna and Martí (2001) as a form of intensification.

Our implementation of path relinking has two phases. In the first one a set of elite solutions is generated with the GRASP method. Instead of retaining only the best solution overall when running GRASP, this phase stores the 10 best solutions obtained with the method. In the second phase we apply the relinking process to each pair of solutions in the elite set. Given the pair (A,B), we consider two paths: from A to B (where A is the *initiating solution* and B the *guiding one*), and from B to A (where they interchange their roles).

The relinking process implemented in our search may be summarized as follows: Let C be the candidate list of vertices to be examined. At each step, a vertex v is chosen from C and labeled in the *initiating solution* with its label *g(v)* in the *guiding solution*. To do this, we look in the *initiating solution* for the vertex u with label *g(v)* and perform *move(u,v),* then vertex v is removed from C. The candidate set C is initialized with a randomly selected vertex. In subsequent iterations, each time a vertex is selected and removed from C, its adjacent vertices are included in C.

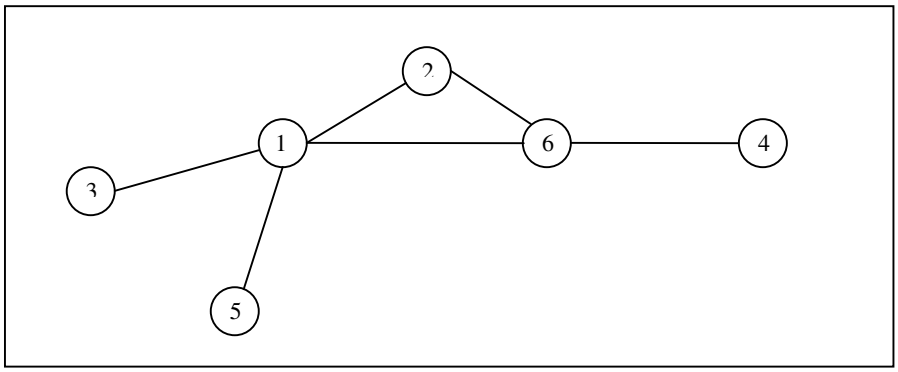

Figure 3. Path Relinking Example

The example in Figure 3 illustrates the path relinking process. Consider the following two solutions A and B and the path from A to B.

Solution A

Node	1	2	3	4	5	6
Label	4	2	6	1	3	5

Solution B

1	2	3	4	5	6
3	1	5	6	2	4

We start with vertex 1 in solution A (C={1}). Since the label of this vertex is 4 in A and 3 in B, we then exchange labels 3 and 4 in solution A corresponding to the vertices 5 and 1, respectively. So, solution A has changed to A_1 (i.e. we have performed the move from A to A_1).

Solution A_1

Node	1	2	3	4	5	6
Label	3	2	6	1	4	5

Solution A_2

1	2	3	4	5	6
3	1	6	2	4	5

Now, we remove vertex 1 from C and add the adjacent vertices of vertex 1: C={2, 3, 5, 6}. The label of vertex 2 in solution A_1 is 2 and in solution B is 1. Label 1 is assigned to vertex 4 in solution A_1, and therefore we exchange labels 2 and 1 in solution A_1, corresponding to vertices 2 and 4, respectively. We obtain solution A_2. In this way the relinking process continues up to matching solution B in 5 steps.

The GRASP_PR procedure starts with the creation of an initial elite set of solutions. The GRASP method is used to build a large set of solutions from which the *n_elite* bests are included in the elite set. The relinking process is then initiated by exploring all pairs of solutions in the elite set. The pairs are selected one at a time in lexicographical order. Each time the relinking process produces a solution that is better than the worst in the elite set, it is replaced. The updating of the elite set is based on improving the quality of the worst solution and the GRASP_PR procedure terminates when no new solutions are admitted to the elite set. We have considered two updating strategies for the elite set during preliminary experimentation. The *dynamic strategy* adds a new solution to the elite set if it qualifies as soon as it is generated in a path. The *static strategy* stores the new solutions in an intermediate set and updates the elite set once it has been fully explored. The dynamic strategy is more aggressive while the static explores all the original solutions in the elite set. Comparisons of both strategies are reported in the next Section.

We have experimentally found that in most cases this relinking process by itself does not produce better solutions than the initiating and guiding solutions. It is convenient to add a local search exploration from some of the visited solutions in order to produce improved outcomes. These results are in line with those reported in Laguna and Martí (1999) for the arc crossing problem. Specifically, we have applied the local search method introduced in the previous section to some of the solutions generated in the path. Note that two consecutive solutions after a relinking step differ only in the label of two vertices. Therefore, it does not seem efficient to apply the local search exploration at every step of the relinking process. We introduce the parameter *n_improves* to control the application of the exchange mechanism. In particular, the exchange mechanism is applied *n_improves* times in the relinking process. We report on the effectiveness of the procedure with different values of this parameter in the computational testing that follows.

5. Computational Experiments

The procedures described in the previous section were implemented in C, and all experiments were performed on a K7-Athlon PC at 1200 MHz. We have also considered the GPS and Tabu search implementations as they appear in Martí et al. (2001) as well as the set of 113 instances from the *Harwell-Boeing Sparse Matrix*

Collection. The codes were compiled with Microsoft Visual C++ 6.0, optimising for maximum speed.

Before testing the effectiveness of our procedures, we perform two preliminary experiments to compare move selection strategies and to explore the effect of changes in the path relinking parameters. We consider 30 representative problem instances with the goal of finding appropriate values for the key search parameters.

Preliminary Experiment I compares the *best* versus the *first* strategy for move selection in the construction phase of the GRASP method, as described in Section 2. Table 2 shows the average objective function value, $B_f(G)$, the average percentage deviation from the best solution found with both strategies, the average running time and the number of best solutions that each strategy was able to find. By comparing the results of Table 2, it is clear that the first strategy provides slightly better results than the best one in significant shorter CPU time. Therefore, we select the *first* strategy for the rest of the experiments.

Table 2. Preliminary Experiment I

	Best	**First**
$B_f(G)$	82.17	81.80
Deviation	0.3%	0.1%
CPU seconds	81.1	59.4
No. of minima	25	28

Preliminary Experiment II has the goal of finding appropriate values for the three critical path relinking search parameters: *Strategy* (of updating the elite set), *n_elite* and *n_improves*. For this purpose, we employ a full factorial design with the parameter values given in Table 3. The 12 tests resulted in the best setting of *Strategy*=Static, *n_elite*=10, and *n_improves*=20.

Table 3. Preliminary Experiment II

Strategy	Dynamic, Static
n_elite	5, 10
n_improves	10, 15, 20

With the search parameters set as indicated above, we proceed to compare the relative merit of our GRASP variants. The GRASP method is run for 200 iterations. Table 4 shows, for each method, the average bandwidth over the instances in each set along with the average CPU seconds. This table also shows the average deviation from the best-known solutions. The best-known solutions are the best solutions found by applying all the procedures to the same problem instance. We cannot assess how close the best are to the optimal solutions, and we are only using these values as a way of comparing the methods.

Table 4. Performance comparison according to problem size

	GPS	**TS**	**GRASP**	**GRASP_PR**
33 instances with n=30. 199				
$B_f(G)$	31.42	23.33	23.58	22.52
Deviation	32.39%	7.28%	4.17%	0.14%
CPU seconds	0.003	4.99	0.33	4.03
80 instances with n=200. 1000				
$B_f(G)$	156.38	100.78	107.56	99.43
Deviation	41.98%	8.07%	11.20%	2.97%
CPU seconds	0.11	263.97	128.47	339.97

Table 4 shows that the performance of the GPS approach is clearly inferior, with average deviations several orders of magnitude larger than those obtained with the other methods. On the other hand, TS, GRASP and GRASP_PR cannot compete with GPS in terms of CPU time.

The GRASP procedure outperforms the TS method in small instances. The average deviation from the best known values is 4.17% for the GRASP, while TS obtains an average deviation of 7.28%, using much more running time (i.e., 0.3 seconds for GRASP versus 4.9 seconds for TS). In large instances, the TS method obtains better solutions than the GRASP, although it employs longer running times. Also note the remarkable improvement of GRASP when the path relinking is used (GRASP_PR), with the average deviation decreasing from 4.17% to 0.14% in small instances and from 11.20% to 2.97% in large instances. The best solution quality is obtained by the GRASP_PR method, which is able to match 81 out of the 113 best solutions known (which compares favorably with TS that is only able to find 41 best solutions).

The next experiment has the goal of showing how the average solution obtained by TS, GRASP and GRASP_PR improves over the time. The results of this experiment are shown in Figure 4 where the procedures were run for 255 seconds. Note that in the first 100 seconds, GRASP is able to obtain better solutions than TS, which needs more than 175 seconds to obtain solutions of similar quality. Comparing GRASP with GRASP_PR, it is clear that if more than 100 seconds are available, it is better to apply the path relinking process to the best solutions obtained (such as the GRASP_PR), instead of continuing to generate solutions as the GRASP method does.

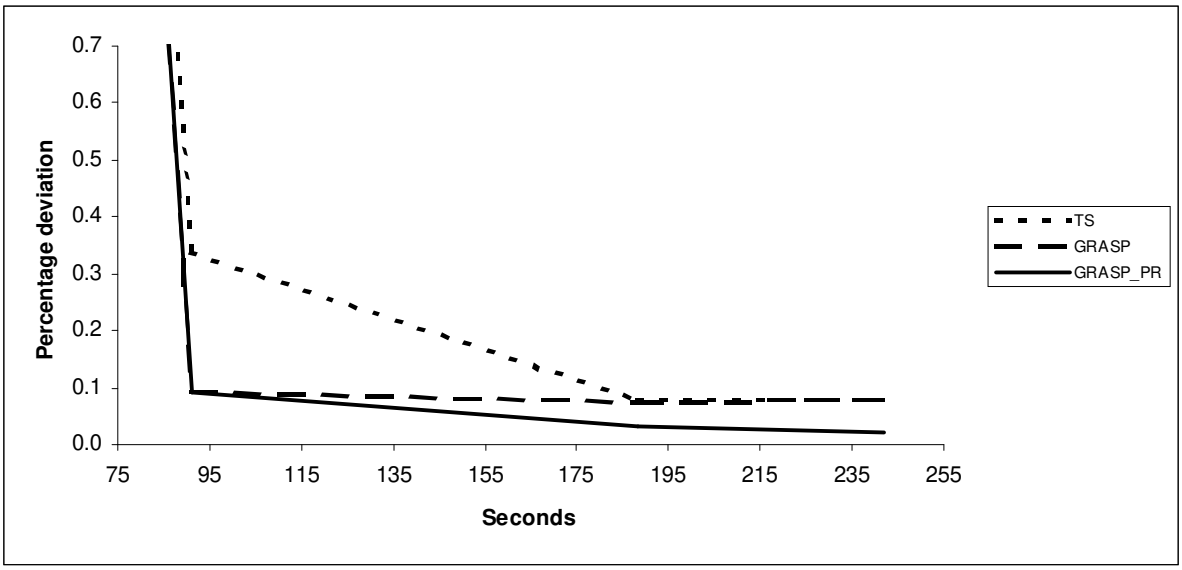

Figure 4. Performance graph for three procedures

In our last experiment we compare the performance of the TS and GRASP_PR procedures on smaller graphs (as compared to those in the previous experiments). In specific, we generate 98 additional sparse graphs with the number of vertices ranging from 15 to 24 and the number of edges ranging from 17 to 49 (a similar density to the instances of the previous experiments). We target small instances in order to compute the optimal solutions by solving an integer programming problem. We propose the following formulation for the MBMP:

$$(MBMP) \quad Min\ b$$

$$s.a.:$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \in V \qquad (1)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j \in V \qquad (2)$$

$$\sum_{k=1}^{n} k x_{ik} = l_i \quad \forall i \in V \qquad (3)$$

$$\left.\begin{array}{l} b \geq l_i - l_j \\ b \geq l_j - l_i \end{array}\right\} \quad \forall \{i, j\} \in E \qquad (4)$$

$$x_{ij} \in \{0,1\} \quad \forall i, j \in V \qquad (5)$$

In this formulation, $x_{ij} = 1$ if label j is assigned to vertex i, and 0 otherwise. Constraints (1) and (2) ensure that each vertex has a label and each label is assigned to a vertex. Constraints (3) set the labels to the vertices according to the assignment variables x_{ij}; so if the variable $x_{ij} = 1$, vertex i takes the label $l_i = j$. Constraints (4) guarantee that the bandwidth b is greater or equal to the absolute difference between the labels of all pairs of adjacent vertices. Finally, the objective function is to minimize the bandwidth of the graph. The problems were solved with CPLEX 7.0. The time required ranged from a minimum of a few seconds to a maximum of more than 50 hours.

In this experiment, TS is able to match 71 optima while GRASP_PR obtains 94 out of 98 (both present similar running times). We perform an additional experiment where both procedures run until the optimum is reached. The TS method takes 2.92 seconds on average while GRASP_PR takes 0.03 to obtain all the optimal solutions. However, if we remove one outlier instance where the TS presents an extraordinarily long running time, the average running time of the TS is reduced to 1.28, while that presented by the GRASP_PR remains the same.

6. Conclusions

We have developed a heuristic procedure based on the GRASP methodology to provide high quality solutions to the problem of minimizing the bandwidth of a graph (matrix). We have explored the critical issue of which solution-generation-method proves effective to obtain a good set of solutions in terms of quality and diversity. The GRASP procedure may be coupled with a path relinking strategy to search for improved outcomes. Unlike other metaheuristics, path relinking has not yet been extensively studied. In particular, we have undertaken to examine the adaptation of path relinking in the context of Multi-Start methods such as GRASP.

Overall experiments with 211 instances were performed to assess the merit of the procedures developed here. Our implementation was shown to be competitive in a set of instances previously reported in the literature. The procedure has been shown to be robust in terms of solution quality within a reasonable computational effort. The proposed method was compared with a recently developed tabu search procedure (Martí et al., 2001). The comparison favours the proposed GRASP_PR implementation.

References

Cuthill, E. and J. McKee (1969) "Reducing the Bandwidth of Sparse Symmetric Matrices," *Proc. ACM National Conference*, Association for Computing Machinery, New York, pp. 157-172.

Dueck, G. H. and J. Jeffs (1995) "A Heuristic Bandwidth Reduction Algorithm," *J. of Combinatorial Math. And Comp.*, vol. 18, pp. 97-108.

Duff, I. S., R. G. Grimes and J. G. Lewis (1992) "Users' Guide for the Harwell-Boeing Sparse Matrix Collection," Research and Technology Division, Boeing Computer Services.

Feo, T. and M. G. C. Resende (1995) "Greedy Randomized Adaptive Search Procedures," *Journal of Global Optimization*, vol. 2, pp. 1-27.

Gibbs, N. E., W. G. Poole and P. K. Stockmeyer (1976a) "An Algorithm for Reducing the Bandwidth and Profile of Sparse Matrix," *SIAM Journal of Numerical Analysis*, vol. 13, no. 2, pp. 236-250.

Gibbs, N. E., W. G. Poole and P. K. Stockmeyer (1976b) "A Comparison of Several Bandwidth and Profile Reduction Algorithms", *ACM Transactions on Mathematical Software*, vol. 2, no. 4, pp. 322-330.

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

Laguna, M. and Martí, R. (1999), GRASP and Path Relinking for 2-Layer straight line crossing minimization", INFORMS Journal on Computing, vol. 11 (1), pp. 44 – 52.

Luo, J.C. (1992) "Algorithms for Reducing the Bandwidth and Profile of a Sparse Matrix", *Computers and Structures*, vol. 44, pp. 535-548.

Martí, R., Laguna, M., Glover, F. and Campos, V. (2001) "Reducing the Bandwith of a Sparse Matrix with Tabu Search", *European Journal of Operational Research*, 135(2), pp. 211-220.