New Ideas and Applications of Scatter Search and Path Relinking

FRED GLOVER[*], MANUEL LAGUNA

Leeds School of Business, University of Colorado,
Boulder, CO 80309-0419, USA. Fred.Glover@Colorado.edu.
Manuel.Laguna@Colorado.edu

RAFAEL MARTÍ[*]

Departamento de Estadística e Investigación Operativa
Universitat de València, 46100 Burjassot, Spain. Rafael.Marti@uv.es

Latest version: December 4, 2002

Abstract — Practical elements of Scatter Search and Path Relinking are illustrated by seven recent applications. The computational outcomes, based on comparative tests involving real world and experimental benchmark problems, demonstrate that these methods provide useful alternatives to more established search procedures. The designs in these applications are straightforward, and can be readily adapted to other optimization problems of varied structures.

[*] Research partially supported by the Office of Naval Research Contract N00014-01-1-0917 in connection with the Hearin Center of Enterprise Science at the University of Mississippi.

[*] Partially supported by the *Ministerio de Ciencia y Tecnología* of Spain (Ref.TIC2000-1750-C06-01)

1. Introduction

Scatter Search (SS) and Path Relinking (PR) have recently been investigated in a number of studies. In this chapter we disclose some of the practical performance aspects of these methods by examining the following seven recent applications:

a. Neural Network Training
b. Multi-Objective Routing Problem
c. OptQuest: A Commercial Implementation
d. A Context-Independent Method for Permutation Problems
e. Classical Vehicle Routing
f. Matrix Bandwidth Minimization
g. Arc Crossing Minimization

The designs in these applications are straightforward, and can be readily adapted to other optimization problems of similarly diverse structures.

SS and PR may be viewed as evolutionary algorithms that construct solutions by combining others, and derive their foundations from strategies originally proposed for combining decision rules and constraints (Glover, 1963, 1965). Chapter 1 describes the fundamental principles and processes underlying these methodologies. We limit our attention here to sketching specific applications that demonstrate the scope and impact of these procedures.

2. Scatter Search Applications

The descriptions that follow are edited versions of reports by researchers and practitioners who are responsible for the applications of SS cited in this section. Definitions of terms and basic procedural components employed are taken from Chapter 1.

2.1 Neural Network Training

A highly effective adaptation of Scatter Search to the neural network training problem has been developed by Laguna and Martí (2000). The improvement procedure embedded in the SS method consists in this case of the well known Nelder and Mead (1965) Simplex method for unconstrained nonlinear optimization. Given a set of weights *w*, the Simplex method starts by perturbing each weight to create an initial simplex from which to begin the local search. The algorithm uses the implementation of the Nelder-Mead method described in Press, et al. (1992).

The SS procedure itself begins by generating the appropriate data normalizations, and then creates an initial reference set (*RefSet*) of *b* solutions. A set *P* of *PSize* solutions (bounded between *wlow* and *whigh*) is built with the diversification method, based on a controlled randomization scheme, given in Glover, Laguna and Martí (1999). *RefSet* is filled with the best *b*/2 solutions in *P* that result by applying the improvement method.Then *b*/2 additional solutions are generated as perturbations of the first *b*/2 and are added to *RefSet*. The perturbation consists of multiplying each weight by 1 + U[-0.05,0.05], where U is the uniform distribution.

In step 2, the solutions in *RefSet* are ordered according to quality, where the best solution is the first one in the list. Then, the *NewPairs* set is constructed consisting of all the new pairs of solutions that can be obtained from *RefSet*, where a "new pair" contains at least one new solution. The pairs in *NewPairs* are selected one at a time to create linear combinations. The improvement method is applied to the best *b* solutions created as linear combinations. Each improved solution is then tested for admission into *RefSet*. If a newly created solution improves upon the worst solution currently in *RefSet*, the new solution replaces the worst and *RefSet* is reordered.

In step 3 the procedure intensifies the search around the best-known solution. At each intensification iteration, the best-known solution is perturbed (multiplied by 1 + U[-0.05,0.05]) and the improvement method is applied. The best solution is updated if the perturbation plus the improvement generates a better solution. After *IntLimit* intensification iterations without improving the best solution, the procedure abandons the intensification phase and returns to step 2. Previously, the improvement method is applied to the best *b*/2 solutions in *RefSet* and the worst *b*/2 solutions are replaced with perturbations of the best *b*/2 (now, each improved solution is multiplied by 1+U[-0.01,0.01]). The training procedure stops when the number of objective function evaluations reaches the total allowed. Preliminary experimentation determined that reasonable values for the parameters *wlow, whigh, b* and *IntLimit* are –2, 2, 10 and 20 respectively.

Computational experiments on the neural network training problem, applied to benchmark problems previously reported in the literature, show that the scatter search implementation compares very favorably with the best known methods for these problems (which include simulated annealing, tabu search, and genetic algorithms). SS reaches a prediction accuracy that that makes it possible to filter out potentially bad solutions generated during the optimization of a simulation, and does so within a computational time that is practical for on-line training.

2.2 Multi-Objective Routing Problem

Corberán et al. (2001) address the problem of routing school buses in a rural area. The authors approach this problem with a node routing model with multiple objectives that arise from conflicting viewpoints. From the point of view of cost, it is desirable to minimize the number of buses (*m*) used to transport students from their homes to school and back. And from the point of view of service, it is desirable to minimize the time that a given student spends in route. The current literature deals primarily with single-objective problems and the models with multiple objectives typically employ a weighted function to combine the objectives into a single one.

The solution procedure considers each objective separately and search for a set of efficient solutions instead of a single optimum. The SS approach for constructing, improving and then combining solutions consists of the following elements:

H1 and H2:	Two constructive heuristics to generate routes
SWAP:	An exchange procedure to find a local optimal value for the length of each route
INSERT:	An exchange procedure to improve upon the value of *tmax,* which identifies the maximum time in the bus.
COMBINE:	A mechanism to combine solutions in a reference set of solutions in order to generate new ones.

The overall procedure operates as follows. (See the outline in Figure 1.) The constructive heuristics H1 and H2 are applied with several values for *tmax* and the resulting solutions are stored in separate pools, one for each value of *m*. The larger the value of *tmax* the larger the frequency in which the heuristics construct solutions with a small number of routes. Conversely, solutions with a large number of routes are obtained when the value of *tmax* is decreased. The procedure then attempts to improve upon the solutions constructed by H1 and H2. The improvement consists of first applying SWAP to each route and then applying INSERT to the entire solution. If any route is changed during the application of INSERT then we apply SWAP one more time to all the changed routes. The procedure now iterates within a main loop, in which a search is launched for solutions with a common number of routes. The main loop terminates when all the *m*-values have been explored.

From all the solutions with *m* routes, the best *b* are chosen to initialize the reference set (*RefSet*). The criterion for ranking the solutions at this step is *tmax*, since all solutions have the same number of routes. The procedure performs iterations in an inner-loop that consists of searching for a solution with *m* routes with an improved $t_{\max}$ value. The combination procedure COMBINE is applied to all pairs of solutions in the current reference set *RefSet*. The combined solutions are improved in the same way as described above, that is, by applying SWAP then INSERT and finally SWAP to the routes that changed during the application of INSERT. We refer to the resulting set of distinct solutions as *ImpSet*. The reference set is then updated by selecting the best *b* solutions from the union of *RefSet* and *ImpSet*. Steps 5, 6 and 7 in the outline of Figure 1 are performed as long as at least one new solution is admitted in the reference set.

```
1. Construct solutions — Apply constructions heuristics
   H1 and H2 with several values of TMAX.
2. Improve solutions — Apply SWAP to each route in a
   solution and INSERT to the entire solution. Finally,
   apply SWAP o any route changed during the
   application of INSERT.
3. Build solution pools — Put all solutions with the same
   number of routes in the same pool.
for ( each solution pool ) do
    4. Build the reference set — Choose the best b
       solutions in the pool to build the initial RefSet.
    while ( new solutions in RefSet ) do
        5. Combine solutions — Generate all the
           combined solutions from pairs of reference
           solutions where at least one solution in the
           pair is new.
        6. Improve solutions — Apply SWAP to each
           route in a solution and INSERT to the entire
           solution. Finally, apply SWAP o any route
           changed during the application of INSERT.
        7. Update reference set — Choose the best b
           solutions from the union of the current
           reference set and the combined-improved
           solutions to update the RefSet.
    end while
end for
```

Figure 1. SS for Multi-Objective Vehicle Routing

After the reference set is updated, the combination procedure may be applied to the same solution pairs more than once. Since the combination procedure includes some randomized elements, the combination of two solutions may result in a different outcome every time COMBINE is applied. Also, the size of the reference set is increased if the updating procedure fails to add at least one new solution. The additional solutions come from the original pool of solutions generated with the construction heuristics. The reference set size is increased up to 2**b*, where *b* is the initial size.

The computational testing on a real-world problem with 42 primary (elementary) schools and 16 (middle) secondary schools in the Province of Burgos (Spain), reveals the procedure is capable of generating a highly effective approximation of the efficient frontier for each routing problem. Decision makers may use efficient solutions to estimate the best service level (given by the maximum route length) that can be obtained with each level of investment (given by the number of buses used). The results show that several of the solutions implemented in practice are not efficient and can be improved by the SS methodology of the study.

2.3 OptQuest: A Commercial Implementation

OptQuest, a registered trademark of OptTek Systems Inc., is commercial software designed for optimizing complex systems, such as those formulated as simulation models. Many real world optimization problems in business, engineering and science are too complex to be given tractable mathematical formulations. Multiple non-linearities, combinatorial relationships and uncertainties often render challenging practical problems inaccessible to modeling except by resorting to more comprehensive tools (like computer simulation). Classical optimization methods encounter grave difficulties when dealing with the optimization problems that arise in the context of complex systems. In some instances, recourse has been made to itemizing a series of scenarios in the hope that at least one will give an acceptable solution. Due to the limitations of this approach, a long-standing research goal has been to create a way to guide a series of complex evaluations to produce high quality solutions, in the absence of tractable mathematical structures. (In the context of optimizing simulations, a “complex evaluation” refers to the execution of a simulation model.)

The OptQuest Callable Library (OCL) is designed to search for optimal solutions to the following class of optimization problems:

$$\begin{array}{lll} \text{Max or Min} & F(x,y) & \\ & & \\ \text{Subject to} & Ax \leq b & \text{(Constraints)} \\ & g_l \leq G(x,y) \leq g_u & \text{(Requirements)} \\ & l \leq x \leq u & \text{(Bounds)} \\ & y = \text{alldifferent} & \end{array}$$

where x can be continuous or discrete with an arbitrary step size and y represents a permutation.

In a general-purpose optimizer such as OCL, it is desirable to separate the solution procedure from the complex system to be optimized. The disadvantage of this “black box” approach is that the optimization procedure is generic and has no knowledge of the process employed to perform evaluations inside of the box and therefore does not use any problem-specific information. The main advantage, on the other hand, is that the same optimizer can be applied to complex systems in many different settings. The optimization procedure uses the outputs from the system evaluator, which measures the merit of the inputs that were fed into the model. On the basis of both current and past evaluations, the optimization procedure decides upon a new set of input values (see Figure 2). The optimization procedure is designed to carry out a special “strategic search,” where the successively generated inputs produce varying evaluations, not all of them improving, but which over time provide a highly efficient trajectory to the best solutions. The process continues until an appropriate termination criterion is satisfied (usually based on the user’s preference for the amount of time to be devoted to the search).

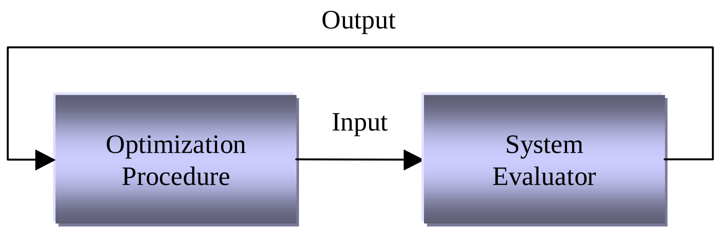

Figure 2. Coordination between optimization and system evaluation

It is assumed that the user has a system evaluator that, given a set of input values, it returns a set of output values that can used to guide a search. For example, the evaluator may have the form of a computer simulation that, given the values of a set of decision variables, it returns the value of one or more performance measures (that define the objective function and possibly a set of requirements). The user-written application uses OCL functions to define an optimization problem and launch a search for the optimal values of the decision variables.

The scatter search method implemented in OCL begins by generating a starting set of diverse points. This is accomplished by dividing the range of each variable into 4 sub-ranges of equal size. Then, a solution is constructed in two steps. First, a sub-range is randomly selected. The probability of selecting a sub-range is inversely proportional to its frequency count (which keeps track of the number of times the sub-range has been selected). Second, a value is randomly chosen from the selected sub-range.

A subset of diverse points is chosen as members of the reference set. A set of points is considered diverse if its elements are "significantly" different from one another. OCL uses a Euclidean distance measure to determine how "close" a potential new point is from the points already in the reference set, in order to decide whether the point is included or discarded.

When the optimization model includes discrete variables, a rounding procedure is used to map fractional values to discrete values. When the model includes linear constraints newly created points are subjected to a feasibility test before they are sent to the evaluator (i.e., before the objective function value F(x) and the requirements G(x) are evaluated). If the solution is infeasible with respect to one or more constraints, OCL formulates and solves a linear programming (LP) problem. The LP (or mixed-integer program, when x contains discrete variables) has the goal of finding a feasible solution x^* that minimizes a deviation between x and x^*.

Once the reference set has been created, a combination method is applied to initiate the search for optimal solutions. The method consists of finding linear combinations of reference solutions. The number of solutions created from the linear combination of two reference solutions depends on the quality of the solutions being combined.

In the process of searching for a global optimum, the combination method may not be able to generate solutions of enough quality to become members of the reference set. If the reference set does not change and all the combinations of solutions have been explored, a diversification step is triggered. This step consists of rebuilding the reference set to create a balance between solution quality and diversity. To preserve quality, a small set of the best (*elite*) solutions in the current reference set is used to seed the new reference set. The remaining solutions are eliminated from the reference set. Then, the diversification generation method is used to repopulate the reference set with solutions that are diverse with respect to the elite set. This reference set is used as the starting point for a new round of combinations.

In Laguna and Martí (2002), the functionality of the library is illustrated with an example in the context of nonlinear optimization. The authors tested OCL by comparing its performance with Genocop III, a third-generation genetic algorithm. Experiments with 30 nonlinear optimization problems show that OCL is a search method that is both aggressive and robust, finding high-quality solutions early in the search and continuing to improve upon the best solution when allowed to search longer. The quality of solutions obtained by OCL uniformly dominated that of solutions obtained by Genocop III, with marked superiority on the more difficult problems. In addition, OCL obtained these improved solutions with speeds ranging from one to three orders of magnitude faster than the genetic algorithm approach. These characteristics make OCL especially useful for applications in which the evaluation of the objective function requires a non-trivial computational effort. OCL has now been used to solve complex optimization problems in more than 20,000 real world applications. More details can be found on the website www.opttek.com .

2.4 A Context-Independent Method for Permutation Problems

Campos, Laguna and Martí (2001) develop a context-independent method for solving problems whose solutions can be represented with a permutation. As in the case of OCL, described in the previous section, this general-purpose heuristic is based on a model that treats the objective function evaluation as a black box, making the search algorithm context-independent. The procedure is a scatter search/tabu search hybrid. The scatter search framework provides a means for diversifying the search throughout the exploration of the permutation solution space. Two improvement methods are used to intensify the search in promising regions of the solution space: a simple local search based on exchange moves and a short-term memory tabu search. Improved solutions are then used for combination purposes within the scatter search design.

The solver is designed in such a way that the user must specify whether the objective function evaluation is more sensitive to the “absolute” positioning of the elements in the permutation or to their “relative” positioning. Hence, we differentiate between two classes of problems:

A-permutation problems —	for which absolute positioning of the elements is more important
R-permutation problems —	for which relative positioning of the elements is more important

The distance between two permutations $p = (p_1, p_2, ..., p_n)$ and $q = (q_1, q_2, ..., q_n)$ depends on the type of problem being solved. For A-permutation problems, the distance is given by:

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

The distance for R-permutation problems is defined as: $d(p,q)$ = number of times p_{i+1} does not immediately follow p_i in q, for $i = 1, ..., n-1$.

In order to design a context-independent combination methodology that performs well across a wide collection of different problems, the authors propose a set of 10 combination methods from which one is probabilistically selected according to its performance in previous iterations. Solutions in the *RefSet* are ordered according to their objective function value. So, the best solution is in the first one in *RefSet* and the worst is the last one. When a solution obtained with combination method i (referred to as cm$_i$) qualifies to be the j^{th} member of the current *RefSet*, we add b-j+1 to *score*(cm$_i$). Therefore, combination methods that generate good solutions accumulate higher scores and increase proportionally their probability of being selected. Other SS elements in the method follow the standard description given in Chapter 1. Two different solvers are proposed, the first one implements a local search phase as the improvement method, the second one uses a short term memory tabu search as the improvement method.

The performance of the procedure has been assessed using 157 instances of four different permutations problems. Solutions to these problems are naturally represented as permutations: the bandwidth reduction problem (BRP), the linear ordering problem (LOP), the traveling salesman problem (TSP), and a single machine sequencing problem (SMS). Solutions obtained with the scatter search/tabu search method have been compared with the best-known solutions to each problem. The procedure has been shown competitive with methods specifically designed for the LOP and SMS problems. The method also provides reasonable results for TSP problems, although not competitive with those obtained by methods that are customized to exploit the special structure of the TSP. The method also is not highly appropriate for the BRP due to the min-max nature of the objective function calculation associated with this class of problems.

For the permutation problems considered, the method was shown superior to comparable procedures that are commercially available. (In all cases, the method produces substantial improvements). The experimentation shows that context-independent methods can be useful in the context of permutation problems, when the associated objective function is capable of discriminating among solutions in a given neighborhood

2.5 Classical Vehicle Routing

Rego and Leão (2002) identify a general design for solving vehicle routing problems using scatter search that has proved exceptionally effective. The vehicle routing problem (VRP) is a classic application in Combinatorial Optimization that can be defined as follows. Let $G=(V,A)$ be a graph where $V=\{v_0,v_1,\ldots,v_n\}$ is a vertex set, and $A=\{(v_i,v_j)\mid v_i,v_j\in V, i\neq j\}$ is an arc set. Vertex v_0 denotes a *depot*, where a fleet of *m* identical vehicles of capacity *Q* are based, and the remaining vertices $V'=V\setminus\{v_0\}$ represent *n cities* (or client locations). A nonnegative *cost* or distance matrix $C=(c_{ij})$ that satisfies the triangle inequality ($c_{ij}\leq c_{ik}+c_{kj}$) is defined on *A*. It is assumed that $m\in[\underline{m},\overline{m}]$ with $\underline{m}=1$ and $\overline{m}=n-1$. The value of *m* can be a decision variable or can be fixed depending on the application.

Vehicles make pickups or deliveries but not both. With each vertex v_i is associated a quantity q_i $(q_0=0)$ of some goods to be delivered by a vehicle and a service time δ_i $(\delta_0=0)$ required by a vehicle to unload the quantity q_i at v_i. The VRP consists of determining a set of *m* vehicle routes of minimal total cost, starting and ending at a depot v_0, such that every vertex in $v_i\in V'$ is visited exactly once by one vehicle, the total quantity assigned to each route does not exceed the capacity *Q* and the total duration of any vehicle route does not surpass a given bound *D*.

The algorithm implementation is structured into five basic components, following the characteristic scatter search design:

Diversification Generation Method:-- To start with an initial set of trial solutions that differ significantly from each other, the generator of combinatorial objects described in Glover (1997) is used to generate permutations in *n*-vectors where components are all vertices $v_i\in V\setminus\{v_0\}$. For a given permutation *P(h),* each cluster of vertices in a route is obtained by successively assigning a vertex $v_i (i\in P(h))$ to a route R_{h_k} (initially *k=1*) until any of the cumulative values for $Q_k=\sum_{v_i\in R_{h_k}} q_i$ or $D_k=\sum_{(v_i,v_j)\in R_{h_k}}(c_{ij}+\delta_i)$ does not exceed *Q* or *D*, respectively, with the insertion of a new vertex v_{k_j}. As soon as such a cutoff limit is attained a new assignment is created by incrementing *k* by one unit, and the process goes on until all vertices have been assigned.

The result obtained can be viewed as a generalized assignment process that does not rely on the order in which clients are visited, though it ensures that all the initial solutions that can be created are feasible and different (since they derive from distinct permutations). Vehicle routes are then determined by using the stem-and-cycle ejection chain algorithm for the traveling salesman problem described in Rego (1998a).

Improvement Method:-- The improvement method is based on the Flower Ejection Chain (FEC) algorithm described in Rego (1998b). For the purpose of the proposed scatter search algorithm, the original FEC procedure has been modified as in the scatter search phase this improvement method is required to deal with infeasible solutions. The method works in two stages. The first stage is concerned with making the solution feasible while choosing the most favorable move and the second stage is the improvement process that operates only on feasible solutions. The method considers varying penalty factors associated with the problem constraints to drive the search toward the feasible region.

Reference Set Update Method:-- A set of reference solutions is created and maintained as follows. Intensification is achieved by the selection of high-quality solutions (in terms of the objective function value) and diversification is induced by including diverse solutions from the current candidate set *CS*. Thus the reference set *RS* is defined by two distinct subsets *B* and *D*, representing respectively the subsets of high-quality and diverse solutions, hence $RS = B \cup D$. A diversity measure, $d_{ij} = \left|(S_i \cup S_j) \setminus (S_i \cap S_j)\right|$ is used to express the distance between solutions S_i and S_j, identifying the number of edges by which the two solutions differ from each other. Candidate solutions are included in *RS* according to the *Maxmin* criterion that maximizes the minimum distance of each candidate solution to all the solutions currently in the reference set.

Subset Generation Method:-- Subsets of reference solutions are generated to create structured combinations in the next step. The method is typically designed to organize subsets of solutions to cover different promising regions of the solution space. In a spatial representation, the convex-hull of each subset delimits the solution space in subregions containing all possible convex combinations of solutions in the subset. In order to achieve a suitable intensification and diversification of the solution space, three types of subsets are required to be organized:

1) subsets containing only solutions in *B*,
2) subsets with only solutions in *D*, and
3) subsets mixing in solutions in *B* and *D* in different proportions.

Subsets defined by solutions of type 1 are conceived to intensify the search in regions of high-quality solutions while subsets of type 2 are created to diversify the search to unexplored regions. Finally, subsets of type 3 integrate both high-quality and diverse solutions with the aim of exploiting solutions across these two types of subregions.

Solution Combination Method:-- The solution combination method is designed to explore subregions within the convex-hull of the reference set. Solutions consist of vectors of variables x_{ij} representing edges (v_i, v_j). New solutions are generated by weighted linear combinations that are structured by the subsets defined in the last step. In order to restrict the number of solutions only one solution is generated in each subset by a convex linear combination. Nevertheless, the set of the generated edges does not necessarily (and usually does not) represent a feasible graph structure for a VRP solution insofar as it may produce a subgraph containing vertices with a degree different from two. Such subgraphs can be viewed as fragments of solutions (or partial routes). Structural feasibility is restored by either linking vertices of degree 1 directly to the depot or dropping edges with the smallest scores, from among those incident at vertices of degree greater than 2, until the degree of each vertex becomes equal to two. While the resulting subgraphs are feasible for the VRP routing structure, they may not yield a feasible solution in relation to the capacity or route length constraints. This latter form of infeasibility is handled by the improvement method as previously indicated.

Computational testing was performed on a set of 26 standard benchmark instances taken from Christofides, Mingozzi and Toth (1972) and Rochat and Taillard (1995). Comparisons with previous VRP algorithms in the literature show the scatter search algorithm not only is competitive with the best of them across a broad spectrum of problems but is highly robust. For example, in 7 out of the 14 instances from the Christofides, Mingozzi and Toth's testbed, the SS approach obtains a solution that succeeds in matching the best solution previously found by any method. For Rochat and Taillard's instances, the SS algorithm dominates all other methods in all instances. Moreover, the approach offers an additional important advantage. Because the problem constraints are handled separately from the solution generation procedures, and are therefore independent of the problem context, this scatter search design can be directly used to solve other classes of vehicle routing problems by applying any domain-specific (local search) heuristic that is able to start from infeasible solutions.

3. Path Relinking Applications

3.1 Matrix Bandwidth Minimization

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 or large-scale power transmission systems.

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:

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

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 $B_f(G)$.

Piñana et al. (2001) propose a PR implementation for this problem consisting of two phases. The first phase uses a GRASP method to generate an initial set of elite (high quality) solutions. 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 a relinking process is applied to each pair of solutions in the elite set. Given the pair (A,B), two paths are considered: 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 the 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.

In a primitive version, the method employs no improvement procedure, but simply operates on the initial elite set of solutions generated by GRASP method (first building a large set of solutions from which the *n*_best are included in the elite set). The relinking process is then applied to all pairs of solutions in the elite set. Each time the relinking process produces a solution that is better than the worst in the elite set, the worst solution is replaced by the new one. The procedure terminates when no new solutions are admitted to the elite set.

It is shown that in most cases this primitive version (which lacks an improvement method) does not produce better solutions than the initiating and guiding solutions. Upon adding a local search exploration from some of the visited solutions in order to produce improved outcomes, the results are in line with those reported in Laguna and Martí (1999) for the arc crossing problem. Specifically, a local search method is applied to some of the solutions generated in the path. Two consecutive solutions after a relinking step differ only in the label of two vertices and hence it is not efficient to apply the local search exploration at every step of the relinking process. The parameter *n_improves* controls the application of the exchange mechanism. In particular, the exchange mechanism is applied *n_improves* times in the relinking process.

Overall experiments with 211 instances were performed to assess the merit of the procedures. Three methods were considered: the acclaimed GPS approach, a tabu search procedure (Martí et al., 2001) that has previously obtained the best known results for this problem, and the proposed PR method. The experiments reveal that the performance of the GPS approach was clearly inferior, with average deviations several orders of magnitude larger than those obtained with the other methods. The PR procedure outperforms the TS method in small instances. In large instances, the TS method obtains better solutions than the PR, although it employs longer running times. The PR procedure has been shown to be robust in terms of solution quality within a reasonable computational effort.

3.2 Arc Crossing Minimization

Researches in the graph-drawing field have proposed several aesthetic criteria that attempt to capture the meaning of a “good” map of a graph. Although readability may depend on the context and the map’s user, most authors agree that crossing reduction is a fundamental aesthetic criterion in graph drawing. In the context of a 2-layer graph and straight edges, the bipartite drawing problem or BDP consists of ordering the vertices in order to minimize the number of crossings.

A *bipartite graph* $G=(V,E)$ is a simple directed graph where the set of vertices V is partitioned into two subsets, V_1 (the left layer) and V_2 (the right layer) and where $E \subseteq V_1 \times V_2$. The direction of the arcs has no effect on crossings so G is considered to be an undirected graph, the arcs to be edges and denote G by the triple (V_1, V_2, E). Let $n_1 = |V_1|$, $n_2 = |V_2|$, $m = |E|$, and let $N(v) = \{w \in V \mid e = \{v, w\} \in E\}$ denote the set of neighbors of $v \in V$. A solution is completely specified by a permutation π_1 of V_1 and a permutation π_2 of V_2, where $\pi_1(v)$ or $\pi_2(v)$ is the position of v in its corresponding layer.

Laguna and Martí (1999) propose a PR procedure for arc crossing minimization in the context of GRASP. This is the first implementation of PR for the purpose of improving the performance of GRASP (as opposed to accompanying it, as in the study previously cited). In the proposed path relinking implementation, the procedure stores a small set of high quality (elite) solutions to be used for guiding purposes. Specifically, after each GRASP iteration, the resulting solution is compared to the best three solutions found during the search. If the new solution is better than any one in the elite set, the set is updated. Instead of using attributes of all the elite solutions for guiding purposes, one of the elite solutions is randomly selected to serve as a guiding solution during the relinking process. The relinking in this context consists of finding a path between a solution found after an improvement phase and the chosen elite solution. Therefore, the relinking concept has a different interpretation within GRASP, since the solutions found from one GRASP iteration to the next are not linked by a sequence of moves (as in the case of tabu search). The relinking process implemented in our search may be summarized as follows:

The set of elite solutions is constructed during the first three GRASP iterations. Starting with the fourth GRASP iteration, every solution after the improvement phase (called the *initiating solution*) is subject to a relinking process by performing moves that transform the initiating solution into the guiding solution (i.e., the elite solution selected at random). The transformation is relatively simple, at each step, a vertex v is chosen from the initiating solution and is placed in the position occupied by this vertex in the guiding solution. So, if $\pi_1^g(v)$ is the position of vertex v in the guiding solution, then the assignment $\pi_1^i(v) = \pi_1^g(v)$ is made. We assume that an updating of the positions of vertices in V_1 of the initiating solution occurs. After this is done, an expanded neighborhood from the current solution defined by $\pi_1^i(v)$ and $\pi_2^i(v)$ is examined. The expanded neighborhood consists of a sequence of position exchanges of vertices that are one position away from each other, which are performed until no more improvement (with respect to crossing minimization) can be found. Once the expanded neighborhood has been explored, the relinking continues from the solution defined by $\pi_1^i(v)$ and $\pi_2^i(v)$ before the exchanges were made. The relinking finishes when the initiating solution matches the guiding solution, which will occur after $n_1+ n_2$ relinking steps.

Two consecutive solutions after a relinking step differ only in the position of two vertices (after the assignment $\pi_1^i(v) = \pi_1^g(v)$ is made). Therefore, it is not efficient to apply the expanded neighborhood exploration (i.e., the exchange mechanism) at every step of the relinking process. The parameter β is used to control the application of the exchange mechanism, by applying the mechanism every β steps of the relinking process.

Overall, experiments with 3,200 graphs were performed to assess the merit of the procedure. The proposed method is shown competitive in a set of problem instances for which the optimal solutions are known. For a set of sparse instances, the method performed remarkably well (outperforming the best procedures reported in the literature).

References

Campos, V., M. Laguna and R. Martí (2001) “Context-Independent Scatter and Tabu Search for permutation problems”, technical report TR03-2001, University of Valencia. http://matheron.uv.es/investigar/tr03-01.ps.

Corberán, A., E. Fernández, M. Laguna and R. Martí (2001), “Heuristic Solutions to the Problem of Routing School Buses with Multiple Objectives” *Journal of the Operational Research Society*, 53 (4), 427-435.

Christofides, N., A. Mingozzi, P. Toth (1972) “The Vehicle Routing Problem,” *Combinatorial Optimization*, Vol 11, pp 315-338.

Fleurent, C. and F. Glover (1997) "Improved Constructive Multistart Strategies for the Quadratic Assignment Problem Using Adaptive Memory," University of Colorado.

Glover, F. (1963), "Parametric combination of local job shop rules", chapter IV, ONR Research memorandum n. 117, GSIA, Carnegie Mellon University, Pittsburgh, PA.

Glover, F. (1965), "A Multi-Phase Dual algorithm for the zero-one integer programming problem", *Operations Research*, 13, (6), 879.

Glover, F. (1997) "A Template for Scatter Search and Path Relinking," in Lecture Notes in Computer Science, 1363, J.K. Hao, E. Lutton, E. Ronald, M. Schoenauer, D. Snyers (Eds.), pp 13-54.

Laguna, M. and Martí R. (2000) "Neural Network Prediction in a System for Optimizing Simulations," *IIE Transaction on Operations Engineering*, forthcoming.

Laguna, M. and Martí R. (2002) "The OptQuest Callable Library", Optimization Software Class Libraries, Stefan Voss and David L. Woodruff (Eds.) 193-218, Kluwer, 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.

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

Nelder, J. A. and R. Mead (1965) "A Simplex Method for Function Minimization," *Computer Journal*, vol. 7., pp. 308-313.

Piñana, E., I. Plana, V. Campos and R. Martí (2001), "GRASP and Path Relinking for the Matrix Bandwidth Minimization", *European Journal of Operational Research*, forthcoming..

Press, W. H., S. A. Teukolsky, W. T. Vetterling and B. P. Flannery (1992) Numerical Recipes: The Art of Scientific Computing, Cambridge University Press (www.nr.com).
Rego, C., P. Leão (2002) "A Scatter Search for the Vehicle Routing Problem" Research Report HCES-08-02, Hearin Center for Enterprise Science, School of Business Administration, University of Mississippi.

Rego, C. (1998a) "Relaxed Tours and Path Ejections for the Traveling Salesman Problem," in Tabu Search Methods for Optimization, *European Journal of Operational Research*, 106, pp 522-538.

Rego, C. (1998b) "A Subpath Ejection Method for the Vehicle Routing Problem, Management Science," Vol. 44, No 10, pp 1447-1459.

Rochat, Y, E. Taillard (1995) "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing," *Journal of Heuristics*, Vol 1, pp 147-167.