

---

# Ejection Chain and Filter-and-Fan Methods in Combinatorial Optimization

Fred Glover<sup>a\*</sup> and César Rego<sup>b</sup>

*a* University of Colorado, Boulder, CO 80309-0419, USA. [fred.glover@colorado.edu](mailto:fred.glover@colorado.edu)

*b* School of Business Administration, University of Mississippi, University, MS 38677, USA.  
[crego@bus.olemiss.edu](mailto:crego@bus.olemiss.edu)

Latest Revision: October 31, 2006

---

**Abstract** — The design of effective neighborhood structures is fundamental to the performance of local search and metaheuristic algorithms for combinatorial optimization. Significant efforts have been made in the creation of larger and more powerful neighborhoods that are able to explore the solution space more extensively and effectively while keeping computation complexity within acceptable levels. The most important advances in this domain derive from dynamic and adaptive neighborhood constructions originating in ejection chain methods and a special form of a candidate list design that constitutes the core of the filter-and-fan method. The objective of this paper is to lay out the general framework of the ejection chain and filter-and-fan methods and present applications to a number of important combinatorial optimization problems. The features of the methods that make them effective in these applications is expected to provide insights into solving challenging problems in other settings.

---

**Keywords:** combinatorial optimization, metaheuristics, tabu search, local search, neighborhood structures, ejection chains, filter-and-fan.

---

\* Corresponding author.

## 1. Introduction

The metaheuristic area has been the focus of extensive research in the last several years, resulting in methods for solving optimization problems that have had a wide range of successful applications in business, engineering and science. Some of the most significant advances have been the design of sophisticated compound neighborhoods coupled with candidate list strategies. The goal is to provide a framework for exploring the solution space effectively with a modest investment of computational effort.

The definition of an efficient neighborhood structure is important for the performance of any algorithm that iteratively explores the solution space of highly complex and constrained problems that typically arise in practice. Recent studies have shown that compound neighborhood structures, based on effective mechanisms for combining moves, have advantages over simple neighborhoods where a single move is used for the transition from one solution to another.

Important advances have been provided by ejection chain methods (Glover 1991, 1992) and a general class of multi-stream neighborhood search constructions, notably represented by the filter-and-fan method (Glover, 1998; Rego and Glover, 2002). An integral part of exploiting such methods stems from joining them with candidate list strategies to isolate restricted yet effective subsets of moves for consideration at each iteration. Designed properly, such strategies particularly reinforce the intensification and diversification themes of tabu search, and provide fertile ground for the application of learning procedures.

In contrast with other more traditional types of neighborhood constructions, ejection chains and filter-and-fan methods are characterized by properties that make them prototypical examples of what we call *dynamic* and *adaptive* search approaches. Characteristically, these methods generate *compound neighborhood structures*, which encompass successions of interdependent (component) moves, rather than simple moves or sequences of independent moves. These methods are *dynamic* because the number of *component moves* used to compose a compound move is not determined in advance, but rather depends on the *depth* (or level) of the neighborhood where the best trial solution is found, which usually varies from one iteration to another. They are *adaptive* because the type of the neighborhood and the move itself are chosen according to the current state of the search.

We begin by presenting in Sections 2 and 3 the fundamental principles underlying ejection chains and filter-and-fan methods, respectively. In Sections 4 and 5, we review a number of prominent ejection chain and filter-and-fan algorithms to illustrate the application of these methods to different classes of problems and to identify the features responsible for their performance. Practical aspects of these methods are highlighted by examining the applications shown in Table 1.

<b>Ejection Chains</b>	<b>Filter and Fan</b>
Traveling Salesman	Facility Location
Vehicle Routing	Protein Folding
Crew Scheduling	Job Shop Scheduling
Quadratic Assignment	Capacitated Minimum Spanning Tree

Table 1 - Featured applications of ejection chain and filter-and-fan methods

These applications were specially selected to embrace a representative variety of models within each method and to expand on their application to problems of different natures and complexities. Our examination of these areas may be regarded as a focused survey, in that we devote attention primarily to models and applications we have encountered through direct experience, though we undertake to point out related work and studies that provide important contributions to the areas examined.

## 2. Ejection Chains

Ejection Chains are variable depth methods that generate a sequence of interrelated simple (component) moves to create a more complex compound move. There are several types of ejection chains, some structured to induce successive changes in problem variables and others structured to induce changes in particular types of model components (such as nodes and edges of a graph). For the original proposals of the ejection chain framework and foundations we refer the reader to Glover (1991, 1992).

Generally speaking, an ejection chain of  $L$  levels consists of a succession of operations performed on a given set of elements, where the  $k$ -th operation changes the state of one or more elements which are said to be *ejected* in the  $k+1$ th operation. This ejection thereby changes the state of other elements, which then lead to further ejections, until no more operations can be made according to some pre-defined conditions. State-change steps and ejection steps typically alternate, and the options for each depend on the cumulative effect of previous steps (usually, but not necessarily, being influenced by the step immediately preceding). The conditions coordinating the ejection chain process are called *legitimacy conditions*, which are guaranteed by associated *legitimacy restrictions*.

In the ejection chain terminology, the order in which an element appears in the chain determines its *level*. The number of levels  $L$  is the *depth* of the ejection chain. The particular level chosen (from among the  $L$  levels generated to provide a move executed by a local search method) usually varies from one iteration to the next. The total number of levels  $L$  can likewise vary, and hence ejection chains fall within the class of *variable depth methods*. In an ejection chain framework, the solution obtained at each level  $k$  of the chain may not represent a feasible solution but may be transformed into a feasible solution by using a complementary operation called a *trial move*. The objective is to create mechanisms, namely neighborhood structures, allowing one solution to be successfully transformed into another.

More formally, let  $S_i$  be the current solution at iteration  $i$  of the local search method, and let  $e_k, t_k$  be the *ejection move* and the *trial move*, respectively, at a level  $k$  of the chain. A neighborhood search ejection chain process consists of generating a sequence of moves  $e_1, t_1, \dots, e_k, t_k, \dots, e_L, t_L$  on  $S_i$  such that the transition from solution  $S_i$  to  $S_{i+1}$  is given by performing a *compound move*  $e_1, e_2, \dots, e_{k^*}, t_{k^*}$ , where  $k^*$  represents the level associated with the highest quality trial solution visited during the ejection chain construction. (There is no need to save trial solutions at other levels.) In the ejection chain context we use the terms *compound move* and *transition move* interchangeably, to specify the move leading from one solution to another in an iteration of the local search procedure.

The effectiveness of such a procedure depends on the criterion for selecting component moves. More specifically, neighboring solutions obtained by an ejection chain process are created by a succession of embedded neighborhoods that lead to intermediate trial solutions at each level of the chain. However, the evaluation of ejection moves can be

made independently from the evaluation of the trial moves, in which case possible trial moves are only evaluated after performing the ejection move at the same level of the chain. In this variant of the approach, the evaluation of an ejection move  $e_k$  only depends on the cumulative effect of the previous ejection moves,  $e_1, \dots, e_{k-1}$ , and is kept separate from the evaluations of trial solutions encountered along the way. The trial moves are therefore restricted to the function of finding the best trial solution that can be obtained after performing the associated ejection move.

In general, an ejection chain of  $L$  levels can be recursively evaluated by computing the ejection values for these levels and summing them to give the trial value for each level. Consider  $N = \{1, \dots, n\}$  the set of problem elements and denote a legitimate neighborhood for an element  $p \in N$  by  $LN(p)$ , thereby identifying a subset of elements of  $N$  that do not violate the legitimacy restrictions. Also, let  $\varphi(p_k, p)$  and  $\delta(p_k, q)$  be respectively the values of an ejection move and trial move at a level  $k$  of the ejection chain. For the sake of simplification, we assume that the *min* function over each of the ejection and trial move evaluation functions brings in its parameter  $p$  and  $q$  the best elements  $p^*$  and  $q^*$  associated with the best ejection and best trial values found, respectively. A general ejection chain procedure for a minimization objective can be sketched as in Figure 1.

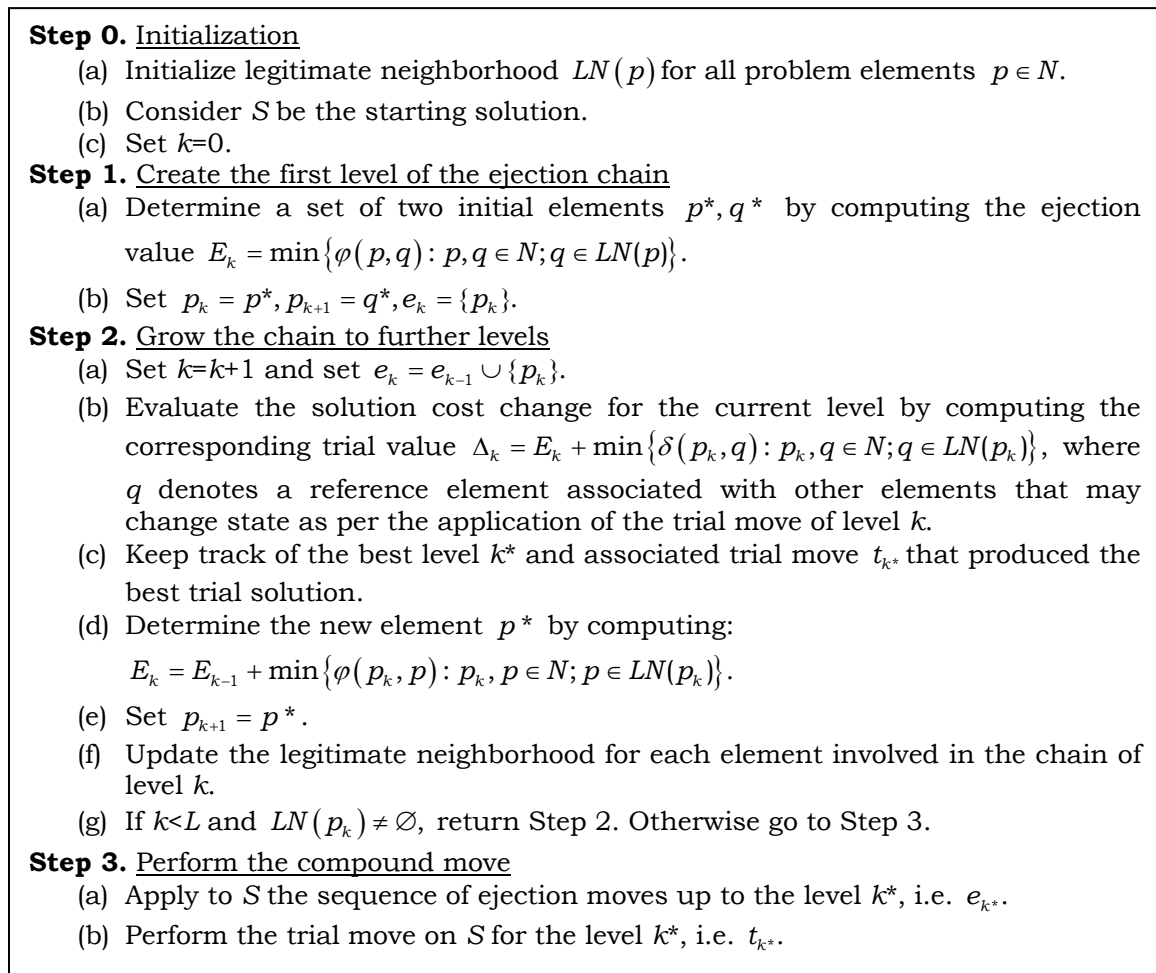


Figure 1 - An iteration of a general ejection chain procedure

We stress that our preceding description of ejection chain processes simply constitutes a taxonomic device for grouping methods that share certain useful features. The value of the taxonomy, however, is evidenced by the role it has played in methods of considerable power for discrete optimization problems across a broad range of applications. As will be seen in the applications selected for this discussion, the ejection chain framework provides a foundation for methods that embrace a variety of compound neighborhood structures with special properties for combining moves, while entailing a relatively modest computational effort.

### 3. Filter and Fan

The filter-and-fan (F&F) method was initially proposed in Glover (1998) as a method for refining solutions obtained by scatter search, and was further extended in Rego and Glover (2002). In the latter, the method is proposed as an alternative to ejection chain methods and as a means for creating combined neighborhood search strategies. Conceptually, it integrates the *filtration* and the *sequential fan* candidate list strategies used in tabu search (Glover and Laguna 1997), and can be viewed as a restrictive form of tabu search that generates multiple paths in a breadth search strategy. From a neighborhood search perspective, the method generates compound moves as a sequence of more elementary *component moves* (or submoves).

Graphically, the F&F model can be illustrated by means of a neighborhood tree where branches represent submoves and nodes identify solutions produced by these moves. An exception is made for the root node, which represents the starting solution to which compound moves are to be applied. The maximum number of levels  $L$  permitted in a single sequence of moves defines the depth of the tree. The neighborhood tree is explored breadth first, level by level. Each level is governed by the *filter candidate list* strategy that selects a subset of moves induced by the *fan candidate list* strategy. The process of selecting moves has to obey a set of *legitimacy conditions* defining associated *legitimacy restrictions* specific to the type of move utilized. The method incorporates two fundamental components: a *local search* to identify a local optimum and a *filter and fan search* to explore larger neighborhoods in order to overcome local optimality. Any time a new local optimum is found in one search strategy the method switches to the other strategy and keeps alternating this way until the filter and fan search fails to improve the current best solution.

#### The Filter and Fan Search

The general F&F search procedure can be sketched as follows. Once a locally optimal solution  $X_0$  is found (in the local search phase) the best  $\eta_1$  currently available moves (among the moves evaluated to establish local optimality) are used to create the level 1 of the F&F neighborhood tree. As a basis for creating the next levels, for a given level indexed by  $k$ ,  $\eta_1$  denotes the number of solutions that are chosen from all solutions available at level  $k$ , as a foundation for generating solutions at level  $k+1$ . (For  $k=1$ , there are just  $\eta_1$  solutions available, so all are chosen.) For each of these  $\eta_1$  solutions, denoted  $X_i(k) (i = 1, \dots, \eta_1)$ , apply  $\eta_2$  moves to generate  $\eta_2$  descendant solutions, thereby generating a total of  $\eta = \eta_1 \cdot \eta_2$  trial solutions for level  $k+1$ . At this stage,  $\eta_1$  of the resulting  $\eta$  solutions are chosen to launch the process for the next level. The values  $\eta_1$  and  $\eta_2$  are input parameters, e.g.  $\eta_1 = 2\eta_2$ . If an improved solution (better than the local optimum  $X_0$ ) is found among the trial solutions, then the method stops branching and switches

back to the local search phase, taking this newly improved solution as a starting point. Otherwise, another selection takes place over the set of moves available.

The process of selecting  $\eta_2$  moves has to obey a set of legitimacy restrictions that assure compatibility of the component moves used for the construction of a valid compound move. The *fan candidate list strategy* is embedded in the generation of the  $\eta$  trial solutions, whereas the selection of the  $\eta_1$  solutions from this collection constitutes the *filter candidate list strategy*.

The basic skeleton of a general F&F procedure is as in Figure 2. Denote  $X^*$  the best solution found so far. Let  $M(k)$  be the candidate list of moves identified at level  $k$  of the F&F tree. Input F&F parameters  $\eta_0, \eta_1, \eta_2$  and let  $L$  be an upper limit for the maximum number of levels of the F&F tree. Build an initial solution  $X$  to initiate the method.

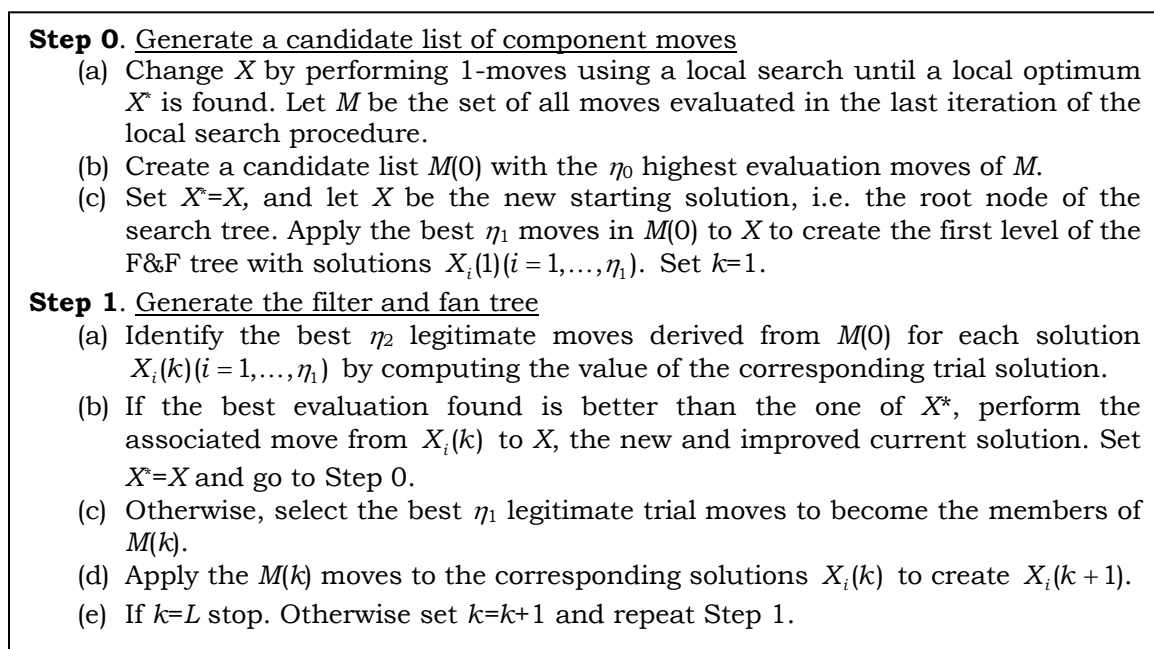


Figure 2 - The general filter-and-fan procedure

In more general versions of the approach, *tree width* and *branch width* can vary adaptively throughout the search in the case where the values for  $\eta_1$  and  $\eta_2$  are changed from level to level. In additional variants of the procedure, as when making use of constructive or destructive neighborhoods, a solution can refer to a partial solution, having some components undetermined. Local optimality is then defined in a special sense relative to the determined components, or by employing a default trial completion that fills in the values of the undetermined components.

More advanced versions allow for the combination of different types of neighborhood structures and the use of adaptive memory programming as introduced in tabu search.

### **Refinements for higher levels of adaptive memory constructions**

To some extent the F&F method can be interpreted as performing multiple threads of tabu searches from the root node of the F&F tree using a limited short-term memory component derived from the legitimacy restrictions. From this perspective a

straightforward enhancement could result by creating a more general algorithm managed by two basic types of short-term memory components: e.g. a *branch-memory* that is local to each branch of the F&F tree or a *tree-memory* that is global to the F&F tree. A limited form of branch-memory is implicitly defined in the legitimacy restrictions of the tree search process. However, the inclusion of more explicit forms of memory would allow different levels of flexibility by using either one of the two indicated types of memory or both memories combined. In that sense a branch-memory could serve to forbid move reversals while tree-memory could be conceived to produce a higher level of diversification of the search among the different branches of the tree.

Higher levels of intensification and diversification can be achieved by incorporating more advanced memory structures as prescribed in tabu search. We conjecture that an effective integration of memories organized at different layers provides a useful means for the creation of the  $M(k)$  candidate lists as well as a vehicle to drive the search in an iterative process that performs the F&F procedure for a number of iterations until a given stopping criterion is met as in general tabu search implementations. Elaborated constructions of the underlying look-ahead process may also be provided by the use of ejection chain processes (performed from nodes at the current level) as a foundation to determine promising component moves to dynamically update the candidate list. Moreover, high evaluation trial solutions found throughout the ejection chain can be recorded for further consideration. All these modifications make recourse to associated elements of tabu search and can directly turn a F&F approach, itself based on the foundations of tabu search, to a higher level tabu search procedure.

## 4. Ejection Chain Applications

### 4.1 Traveling Salesman

The Traveling Salesman Problem (TSP) consists in finding a minimum distance tour of  $n$  cities, starting and ending at the same city and visiting each other city exactly once. In spite of the simplicity of its problem statement, the TSP is remarkably challenging and is the most studied problem in combinatorial optimization, having inspired well over a thousand publications.

In graph theory, the problem can be defined on a graph  $G = (V, A)$ , where  $V = \{v_1, \dots, v_n\}$  is a set of  $n$  vertices (nodes) and  $A = \{(v_i, v_j) \mid v_i, v_j \in V, i \neq j\}$  is a set of arcs, together with a non-negative cost (or distance) matrix  $C = (c_{ij})$  associated with  $A$ . The problem is considered to be symmetric (STSP) if  $c_{ij} = c_{ji}$  for all  $(v_i, v_j) \in A$ , and asymmetric (ATSP) otherwise. Elements of  $A$  are often called edges (rather than arcs) in the symmetric case. The version of STSP in which distances satisfy the triangle inequality ( $c_{ij} + c_{jk} \geq c_{ik}$ ) is the most studied special case of the problem. The STSP (ATSP) consists in determining the Hamiltonian cycle (circuit), often simply called a *tour*, of minimum cost.

The importance of identifying effective heuristics to solve large-scale TSP problems prompted the “8<sup>th</sup> DIMACS Implementation Challenge”, organized by Johnson, McGeogh, Glover, and Rego (2000) and solely dedicated to TSP algorithms.

Ejection chain methods lead the state-of-the-art in local search heuristics for the traveling salesman problem (TSP) and likewise have successfully been applied to a cardinality-constrained variant of the problem (Cao and Glover, 1997). The most effective local search approaches for the classical TSP primarily originate from the

Stem-and-Cycle (S&C) ejection chain method (Glover, 1992) and the widely acclaimed Lin-Kernighan (LK) procedure (Lin and Kernighan, 1973), which can be viewed as an instance of an ejection chain method. These two types of TSP ejection chain approaches typically proceed by disconnecting a subpath and reconnecting it with different components at each level of the chain, and as a consequence are generally called *subpath ejection chain methods*.

### **Subpath ejection chains for the TSP**

Subpath ejection chain methods for the TSP start from an initial tour and iteratively attempt to improve the current solution, generating moves coordinated by a *reference structure*. The LK approach uses a Hamiltonian path as the reference structure to generate moves throughout the neighborhood construction. By contrast, the S&C ejection chain method is based on the stem-and-cycle reference structure. The S&C reference structure is a spanning subgraph of  $G$  consisting of a path called a *stem* connected to a *cycle* by a single node called the *root* node. The two nodes adjacent to the root in the cycle are called *subroots* and the node on the other end of the stem is called the *tip* of the stem. In a subpath ejection chain, once a reference structure is created from the initial TSP tour, ejection moves consist of transforming the reference structure into another of the same type and appropriate trial moves are used to generate feasible tours at each level of the chain.

The LK method starts by generating a low order  $k$ -opt move (with  $k \leq 4$ ) and then creates a Hamiltonian path by deleting an edge adjacent to the last one added. This completes the first level of the LK process. In succeeding levels each ejection move consists of linking a new edge to the unique degree 1 node that was adjacent to the last edge added, followed by deleting the sole edge whose removal will generate another Hamiltonian path. A trial move consists of linking the two endpoints of the current Hamiltonian path, thus creating a feasible tour.

The S&C method starts by creating the initial S&C reference structure from a TSP tour, by linking two nodes of the tour and removing one of the edges adjacent to one of those nodes. Each ejection move links the tip node to any other node on the graph, except for the one adjacent to the tip and removes one of the edges adjacent to that node. Two different ejection moves are possible depending where in the graph the node to be linked to the tip is placed (in the stem or in the cycle). If such node is in the stem there is only one possibility to eject a subpath, which results from deleting the only possible adjacent edge that creates a feasible structure; otherwise two possible subpaths may be ejected by deleting either adjacent edge.

The S&C structure and the nature of its ejection moves are illustrated in Figure 3. In the figure, the S&C structure is represented by dark edges with nodes  $t$ ,  $r$ ,  $s_1$  and  $s_2$  denoting the tip, root and the two subroots of the structure, respectively. Dotted lines denote edges to be added by each type of ejection move and the associated possible edges to be deleted by the move are marked by the small parallel lines crossing them.



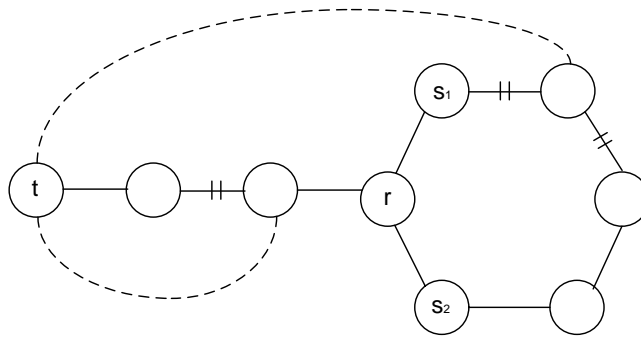


Figure 3 - The S&C reference structure and associated ejection moves

Trial solutions are obtained by adding an edge from the tip to one of the subroots and deleting the edge between this subroot to the root.

Both theoretical and experimental studies have demonstrated that the S&C ejection chain method is more general and powerful than the LK approach. Notably, the reference structure in the LK approach is very close to being a valid TSP solution (it only requires adding a single edge to close the gap between the two nodes of degree 1 and thus obtain a tour). As a result, the structure implicitly limits the different types of moves it can generate and consequently makes only one trial solution available from a given Hamiltonian path. The S&C reference structure, on the other hand, yields two trial solutions (except in the case of a degenerate structure when the tip and root nodes coincide, in which case the structure corresponds to a tour). Another fundamental difference is that the S&C procedure is conceived to generate *dynamic alternating paths* while the classical LK approach generates *static alternating paths*. A theoretical analysis of the differences between the types of paths generated by S&C and LK procedures is provided in Funke, Grünert and Irnich (2005), which includes a demonstration that the LK neighborhood is strictly contained in the S&C neighborhood. The authors also show that even a generalization of the LK approach that incorporates generalized alternating paths cannot reach solutions accessible to the S&C neighborhood.

### The Symmetric TSP

An effective algorithm design and implementation of the S&C ejection chain method was first proposed by Rego (1998a) for the STSP and subsequently enhanced in Gamboa, Rego and Glover (2005, 2006). In the latter, the authors have adopted the *two-level tree* data structure described in Fredman et al. (1995) that is used to support the most efficient LK implementations reported in the DIMACS Challenge (e.g. those of Johnson and McGeoch, 1997; and Helsgaun, 2000; Applegate, Cook, and Rohe, 2003). The upgraded S&C algorithm also incorporates a variety of neighbor lists, thus providing the algorithm with additional options not available in the previous version.

The generation of moves throughout the ejection chain process is based on the definition of a set of rules and legitimacy restrictions on the set of edges that are allowed to be used in subsequent steps of an ejection chain. The algorithm is implemented as a local search improvement method in the sense that no meta-strategy is used to guide the search beyond local optimality. Also, the method always stops after  $n$  iterations of the *re-routing* strategy fail to improve the best solution found so far. (Re-routing consists of starting an S&C ejection chain from a different route node.) This makes our implementation of the S&C algorithm simpler than LK implementations that

make use of additional supplementary techniques such as caching distances, and other implementation tricks.

Maintaining the fundamental rules of the original algorithm (described in Rego, 1998a) unchanged, improvements on the data structures and the use of appropriate candidate list strategies made the modified version of the S&C algorithm more efficient and effective for solving very large-scale problems.

In Gamboa, Rego and Glover (2006) and Gamboa et al. (2006) the authors report the outcomes of an extensive series of tests on problems ranging from 1000 to 3,000,000 nodes, showing that by intelligently exploiting elements of data structures and candidate lists routinely included in state-of-the-art TSP solution software, the S&C algorithm clearly outperforms all implementations of the LK procedure. Specifically, it is shown that S&C approach finds better solutions than all of the leading LK variants for about 70% of the problems tested. Conspicuously, the 70% advantage of the S&C approach refers to a comparison with the most effective variant of the LK procedure. The second best variant of this approach is dominated by the S&C approach in approximately 97% of the problems. Some other variants failed to find even a single solution better than the S&C approach over all 59 problems tested.

### **The Asymmetric TSP**

The S&C is a fundamental structure in a number of other reference structures used in the creation of ejection chain methods. A direct generalization of the S&C reference structure that has special advantages for the ATSP is called the Doubly-Rooted (DR) S&C (Glover 1996), which considers two root nodes instead of one. The doubly rooted structure has two forms: a *bicycle* in which the roots are connected by a single path, joining two cycles, and a *tricycle* in which the two roots are connected by three paths, thereby generating three cycles. In the DR structure the definition of subroot is extended to any node adjacent to a root node, regardless of whether it is in the cycle or in the stem.

Ejection moves consist of adding a new edge linking one of the subroots to an arbitrary node on the graph and deleting the edge between this subroot and the associated root, resulting in the selected arbitrary node as the new root.

The trial solutions available to the doubly-rooted structure are those generated by the union of the trial solutions available to the single-rooted S&C structure obtained by deleting any edge linking a root node to a cycle subroot. Such a subroot becomes the tip of the S&C, while the (root) node that remains with three incident edges becomes the S&C root.

Rego et al. (2006) provide a comparative study of the DR neighborhood structure and the generalized LK neighborhood for the ATSP proposed in Kanellakis and Papadimitriou (1980) and recently used in the current state-of-the-art local search algorithm for the ATSP by Cirasella et al. (2001). Computational experiments on a standard testbed exhibits superior performance for the DR neighborhood over its LK counterpart, revealing that a straightforward implementation of a DR ejection chain algorithm outperforms the best local search algorithms and obtains solutions comparable to those obtained by the current most advanced iterative local search algorithms specially designed for the ATSP, while requiring dramatically smaller computation time.

Out of 28 instances for which results are available for KP, in only 4 instances did the KP algorithm manage to find tours that are slightly better than those found by the DR algorithm. For the remaining 24 instances, the DR algorithm found 3 tours of similar quality and 21 of superior quality compared to those produced by the KP algorithm. In some cases the quality of solutions found by the DR algorithm exceeded that of the KP algorithm by as much as 5.5%. Even more impressive is the performance of the DR algorithm compared to the sophisticated iterative local search variant (iKP) of the basic KP algorithm (Cirasella et al., 2001). Considering the whole set of 47 benchmark instances both iKP and DR algorithms find an equal number of best solutions (28). Among these, a 0.00% gap from optimality is achieved on 9 instances by the iKP algorithm and on 17 instances by the DR algorithm. Also, it appears that the iKP algorithm requires significantly more computational time on average than the DR algorithm. In some cases differences in speed translate in about 2 hours for the iKP compared to less than 50 seconds for the DR algorithm (which finds tours of better quality).

### **Advances on data structures for large STSPs**

The problem of data representation is fundamental to the efficiency of search algorithms for the TSP and particularly important for large STSP instances. The nature of these algorithms necessitates the performance of certain basic tour operations involving subpath reversal and traversal. The computational effort that must be devoted to these operations becomes increasingly pronounced with larger problem instances.

The 2-level tree (Chrobak et al., 1990) has for many years been considered the preeminent choice for representing the tour, retaining that reputation until the recent emergence of the  $k$ -level satellite tree proposed by Osterman and Rego (2003). The 2-level tree divides the tour into approximately  $n^{1/2}$  segments each containing as many nodes as grouped under a parent node, where a doubly linked list is used to connect both segments and client nodes within the segments. A worst case cost of  $O(n^{1/2})$  for tour operations may be achieved with the 2-level tree representation.

The theory of 2-level tree contributes much to the latest developments on TSP data structures. Its effectiveness has been demonstrated by independent implementations due to Fredman et al. (1995), Gamboa, Rego and Glover (2005, 2006) and numerous participants in the DIMACS TSP Challenge (Johnson et al., 2000).

The  $k$ -level satellite tree expands upon the 2-level tree to allow  $k$  levels instead of two. This is accomplished by dividing the tour into segments containing roughly  $n^{1/k}$  nodes each, and the resulting segments are grouped into *parent segments* containing about  $n^{1/k}$  segments each. A fundamental feature of this  $k$ -level satellite tree is the *satellite list* structure, also proposed by Osterman and Rego (2003) as symmetric counterpart of the classical doubly-linked list structure. The satellite list represents a tour without implying a fixed orientation, making it useful for representing symmetric paths or cycles. It can operate in the same capacity as the doubly-linked list and is equally efficient in terms of both memory and computation of *previous* and *next* queries. Because the satellite list avoids a fixed orientation, the subpath reversal operation can be performed in constant time, whereas for the linked list, every pointer associated with nodes in the reversed path in a linked list must be changed to reflect the appropriate orientation. A satellite design for the  $k$ -level tree is important, not only because of subpath reversal, but also because *next* and *previous* queries do not need to access *parent* nodes. The implications of this benefit are tremendous, considering the

frequency of the need for these operations and the fact that the cost of accessing a parent node varies with the problem size when the data structure is designed optimally. See the reference for a detailed description of the  $k$ -level tree and its properties.

It can be shown that with an adequate choice of  $k$ , a path between two client nodes in the tree can be traversed with a complexity of  $O(\log n)$  rather than  $O(n^{1/2})$ . This result indicates that an optimally designed  $k$ -level tree is the most efficient structure proposed to date.

## 4.2 Vehicle Routing

The Vehicle Routing Problem (VRP) is a generic name given to a class of problems in which a set of routes for a fleet of vehicles, based on one or several depots, must be determined for a number of geographically dispersed *cities* or *customers*, subject to side constraints. The problem is central in the fields of transportation, distribution and logistics and provides a general model for a wide range of practical applications.

Let  $G = (V, A)$  be a graph where  $V = \{v_0, v_1, \dots, v_n\}$  is a vertex (or node) set, and  $A = \{(v_i, v_j) \mid v_i, v_j \in V, i \neq j\}$  is an arc (or edge) set. Consider a *depot* to be located at  $v_0$  and let  $V' = V \setminus \{v_0\}$  denote a set of  $n$  cities (or client locations). A non-negative *cost* or *distance* matrix  $C = (c_{ij})$  is associated with every arc of  $A$ . It is assumed that  $m$  identical vehicles are used, each with capacity  $Q$ , and their number is 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  $\delta_i$  ( $\delta_0 = 0$ ) 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  $v_i \in V'$  is visited only once by precisely one vehicle, where the total quantity assigned to each route does not exceed the capacity  $Q$  and the total duration (travel plus service times) of any vehicle route does not surpass a given bound  $D$ . Hence in this context the cost  $c_{ij}$  is taken to be the travel time between the two associated cities.

As discussed earlier, ejection chain methods have been proved very efficient for solving large scale traveling salesman problems. Generalizations of some of these methods have likewise been extended to deal with multiple routes as required in general vehicle routing problems.

### Node-based ejection chains for the VRP

Node-based ejection chain methods derive from extensions of customary single node insertion and exchange neighborhoods that have been found useful in several classes of graph problems including: machine scheduling, clustering, graph-coloring, vertex covering, maximum clique or independent set problems, vehicle routing problems, generalized and quadratic assignment problems, and the traveling salesman problem, just to cite a few.

Typical node insertion (or shift) neighborhoods involve removing a node from one route and inserting it into another, while typical node exchange (or swap) neighborhoods involve interchanging nodes between routes. In neighborhood search, these insertion

and swapping operations are also performed within a given route (instead of across routes) as a way to re-optimize the associated TSP defined over the nodes of this route. Since the worst case complexity of evaluating a single node insertion and node exchange neighborhood is  $O(n^2)$ , creating compound neighborhoods by combinations of these moves requires an effort that grows exponentially with the number of moves considered in combination. More precisely, the best compound neighborhood of  $k$  moves can be generated and evaluated with  $O(n^k)$  effort. Embedding these simple neighborhoods in an ejection chain framework can notably reduce this effort (Glover 1991).

Rego (2001) develops an ejection chain neighborhood for the VRP that implements a *multi-node insertion* move and a *multi-node exchange* move to yield an important form of *combinatorial leverage*. Specifically, the number of moves represented by a level  $k$  neighborhood is multiplicatively greater than the number of moves in a level  $k-1$  neighborhood, but the best move from the neighborhoods at each successive level can be determined by repeating only the effort required to determine a best first level move.

The ejection chain starts by identifying a node pair  $v_i, v_j$  that yields the best (highest evaluation) ejection move that disconnects node  $v_i$  from its current position and inserts it into the position currently occupied by node  $v_j$ . For subsequent levels, ejection moves consist of selecting a new candidate node to be ejected by the previously ejected node, and then repeating until no other legitimate node exists for ejection. Such an ejection process creates an intermediate structure at each level of the chain where the associated ejected node, say  $v_k$  ( $k = j$  for the first level), is temporarily disconnected from the tour. However a trial solution can be obtained by: (1) inserting node  $v_k$  between two nodes  $v_p$  and  $v_q$  and adding an arc linking the original predecessor and successor of  $v_i$  to close the route – a *multi-node insertion move*; or (2) simply by relocating the last ejected node  $v_k$  to occupy the vacant position left by the node  $v_i$  that initiates the chain – a *multi-node exchange move*.

This composite ejection chain neighborhood has been embedded in a tabu search algorithm, named TabuChain, which is designed to use frequency-based adaptive memory and strategic oscillation to allow for temporary violation of the capacity or maximal route duration constraints. Both sequential and parallel versions of the algorithm have been implemented. The parallel version is based on a synchronous model of parallel searches that allows for a more extensive exploration of the solution space than the basic sequential version. Also, different levels of parallelization are used in order to accelerate the search process. One takes advantage of an ejection chain property that permits ejection and trial moves to be evaluated separately by different processors, potentially reducing the time per iteration by half. Another level of parallelization consists of launching separate processes to re-optimize each individual route. The sequential and parallel methods, each in its own category, remain among the most effective algorithms available for the VRP, producing solutions that are on average 0.77% and 0.55% above the best known solutions for the classical fourteen-instance testbed of Christofides, Mingozzi and Toth (1979).

Node-based ejection chain approaches have also been successfully applied to clustering problems by Dorndorf and Pesch (1994). Principles similar to those underlying the node-based ejection chain method discussed for the VRP are developed and explored in Yagiura, Ibaraki and Glover (2004) to provide an effective algorithm for the generalized assignment problem.

## Subpath ejection chains for the VRP

Another type of ejection chain approach for the VRP concerns a subpath ejection chain method proposed in Rego (1998b). A fundamental feature of this method is the *flower reference structure* that generalizes the stem-and-cycle (S&C) reference structure (discussed in Section 4.1) to a multiple routing context. The flower structure is defined as a spanning subgraph of  $G$ , which consists of a path called *stem* attached to multiple cycles representing routes. In the original paper several components of the flower structure are termed differently than their equivalents in the S&C structure; however to facilitate the discussion in this paper we stick with the terms already introduced for the S&C. Therefore, the node that lies on the intersection of a stem and a cycle will be called a *root* and the nodes adjacent to a root will be called *subroots*. Likewise, the node at the opposite end of the stem from the root will be referred to as the *tip* of the stem. In the flower structure the *root* node always identifies the *depot* and hence these two terms may be used interchangeably.

The consideration of multiple cycles in the flower reference structure extends the ejection and trial moves of the stem-and-cycle to encompass a number of other possibilities. Starting from a given VRP solution, the ejection move to create a flower structure may simply delete one of the edges incident to the root (depot), thus transforming a cycle into a stem, which is also a basic move to deal with routes containing a single city. Such a move that only deletes one edge without adding another may be referred here to as a *drop move* to differentiate it from the moves that replace one edge with a new one and so may be called *add-drop moves*. Similarly, a trial move that transforms a flower structure into a VRP solution may simply link the tip directly to the depot to close the route. Such a trial move may be called a *route-creation move*. By contrast, the type of S&C trial move that links the tip node to one of the subroots and deletes the associated edge incident to the root may be called *route-extension move*, since it extends a route to include the clients currently in the stem that is made to join that route. Depending on the type of ejection and trial moves considered for an ejection chain, the number of vehicle routes can vary: the number decreases if the chain starts with an ejection move that deletes an edge incident to the root and then applies a route-extension trial move, whereas the number increases if the chain starts by applying an add-drop move to one of the routes and a route-creation move is used to obtain a new trial solution.

An important feature of the algorithm concerns the choice of the chain starting rules. Since it is possible to create a flower structure from a given VRP solution by deleting one edge without adding another, such a step always results in a cost reduction in relation to the current solution. Moreover, as the longest edges are usually selected to be deleted, this leads to the outcome that the proper add-drop S&C move will rarely be chosen to start the chain. To avoid this situation, the algorithm considers a penalty factor to provide a more appropriate evaluation of the two types of ejection moves. Experimental tests carried out on problems with different characteristics disclosed that randomly varying this penalty within specific intervals (of real values) was highly advantageous. Different tradeoffs can be obtained in evaluating the two types of moves that initiate an ejection chain depending on three ranges of values as follows. For negative values the drop move is highly penalized, hence an add-drop initiating move is performed. If these values are positive and less than 1, initiating drop moves are again penalized in relation to add-drop moves, but not so strongly. Finally, values greater than 1 yield greater penalties for the add-drop initiating moves and hence favor drop moves to be performed.

Although the Flower reference structure preserves the same properties as the S&C structure and so succeeds in generating dynamic alternating paths and cycles, the violation of the alternating path construction that is caused by an ejection chain process in the VRP setting is less restrictive than in the TSP setting. This increases the move options for the VRP, yielding a heuristic advantage. In this setting, periodically limiting the moves to generate an ordinary alternating path rather than a dynamic alternating path turned out to be useful to avoid modifying adjacent edges at the same step of the algorithm. Nevertheless, such a modification was not completely forbidden in order to allow the most promising changes to be carried out. In sum, on one hand it is sometimes desirable not to simultaneously modify two adjacent edges as a means of inducing some degree of diversification; on the other hand it can also sometimes be desirable to allow such a modification to provide some intensification of the search and possibly reach deeper local optima where new best solutions may be found.

The implementation of this subpath ejection chain method relies on a tabu search guidance to prevent the method from generating flower structures already considered at previous levels of the chain. Guidance by tabu search is also used to govern the creation of alternating paths within the context of the legitimacy conditions used in the algorithm, which as in the case of the TSP problem assure that a given solution can be transformed into any other.

To gauge its performance, the Flower algorithm was tested on an extended set of 30 problems from the literature, which include the classical fourteen-instance set of Christofides, Mingozzi, and Toth (1979), three real-world problems taken from Fisher (1994) and twelve instances considered in Taillard (1993) and Rochat and Taillard (1995). The original goal in creating the Flower algorithm was to produce high-quality solutions rapidly rather than striving to find (new) best solutions, and hence no recourse was made to sophisticated forms of TS guidance — in contrast to TabuChain (previously described) and a number of other algorithms in the literature. Comparisons with algorithms sharing a similar goal of rapid convergence reveal that the Flower algorithm is clearly superior to all of them, producing better solutions and also requiring less running time. When compared with other classes of algorithms that make advanced use of metaheuristic guidance, the Flower algorithm compares quite favorably to these as well, especially when good solutions must be found quickly. In particular, the algorithm is very fast in finding solutions that are within the range of 1% of the best known solution.

### **4.3 Crew Scheduling**

The general crew scheduling problem (CSP) can be formulated as seeking the minimum number of crews necessary to cover a set of trips with duties that have to satisfy a number of regulations and operational constraints.

Cavique, Rego and Themido (1999) address a CSP arising in train transportation and develop a *subgraph ejection chain method* embedded in a tabu search algorithm for the solution of the problem. The algorithm relies on the definition of a number of terms generally used in crew scheduling, which can be introduced in the context of the problem at hand.

The set of trips to be performed by each train defines a *timetable*. A *trip* is a one way movement of a train between two *terminal* points, the smallest period (or elementary crew activity) into which the timetable can be divided. A trip has five attributes: train number, starting place and time, finishing place and time. A *block* is a set of all trips

produced by the same train, and the set of consecutive trips in a block, covered by the same crew, is called a *piece of work* (or *piece*). A *block partition* is a set of non-overlapping pieces of work that exactly covers a block. In this application, a complete duty may be formed by one or two pieces of work, a meal break, the report and clear time and a possible reserve period. The set of contractual and operational constraints include specific relief points, bounds on the durations of pieces of work, report and clear times, duty duration, and possible intervals for meal breaks. A duty that satisfies all problem constraints is called a *feasible duty* and a set of feasible duties covering all trips makes up a feasible *schedule*. The objective of the CSP is to find a feasible schedule with a minimum number of crews (duties) needed to operate the train line.

We now undertake the description of the subgraph ejection chain method and associated tabu search procedure. In contrast with the node based and subpath ejection chain methods that concerned the ejection of nodes or subpaths of a graph structure modeling the problem, the present method considers a *subgraph* as the elementary component to be ejected at each level of the ejection chain process. The method explores a specialized *block partition* technique that underlies the formulation of the *maximum cardinality matching problem* (MCMP) of a non-bipartite graph  $G = (P, D)$ . The method is divided into three fundamental procedures: *block partition*, *graph generation*, and *duty achievement*. The block partition procedure, in the first step, divides the blocks into  $k$  feasible pieces of work, creating the node set  $P = (p_1, \dots, p_k)$ . In the second step, the matching graph  $G$  is built by linking pairs of pieces for all possible duties, creating the edge set  $D = \{(p_i, p_j) \mid p_i, p_j \in P\}$ . Finally, in the third step, a MCMP algorithm is applied to find a maximal matching of pieces to create a schedule. In the solution of the MCMP, the matched nodes represent duties with two pieces and the free (or unmatched) nodes are duties with only one piece of work. Under this model the CSP reduces to the problem of finding the block partition that produces a schedule with a minimum number of duties over all possible partitions.

The enormous number of alternatives to partition the set of blocks for a given timetable entails a very large and complex solution space for which effective search algorithms must be designed. The algorithm considers a tabu search approach based on an embedded neighborhood structure that gives rise to a subgraph ejection chain method defined as follows. A neighborhood structure  $N$  is decomposed in two substructures  $N_1$  and  $N_2$ , which separates the neighborhood space into two subsets.  $N_1$  is an intermediate structure responsible for generating a set of new pieces of work that will replace pieces of the current graph  $G_i = (P_i, D_i)$  transforming it into another graph  $G_{i+1} = (P_{i+1}, D_{i+1})$ .  $N_2$  is a structure defining the set of edges in  $G_{i+1}$  associated with feasible duties. The complete neighborhood structure  $N$  is defined by any possible sequence of moves  $e_1, t_1, \dots, e_k, t_k, \dots, e_L, t_L$  such that  $e_k \in N_1$  and  $t_k \in N_2$ , representing an ejection chain of  $L$  levels. Accordingly, the transition from a solution (schedule)  $S_i$  to a solution  $S_{i+1}$  can be obtained by a sequence of moves  $e_1, e_2, \dots, e_{k^*}, t_{k^*}$  with  $e_{k^*}$  and  $t_{k^*}$  denoting the ejection move and trial move, respectively, at level  $k^*$  where the best trial solution was found.

In the algorithm, ejection moves are defined by three types of elementary operations (1) *shift operation*, which shifts the extreme of a piece to the right or to the left, transferring one or more trips between adjacent pieces, (2) *cut operation*, which splits one piece into two pieces, and (3) *merge operation*, which combines two pieces into a single piece. Each of these operations that modify the configuration of certain nodes require deleting (or ejecting) a subgraph involving the modified nodes and certain edges adjacent to them, which thereby entails the creation of another subgraph associated with the new



configuration of the nodes that have been modified by the ejection move. Under this conception, an ejection move at level  $k$  deletes (ejects) a subgraph  $G_i^{k-}$  of  $G_i^k$  ( $G_i^1 = G_i$ ) and adds another subgraph  $G_i^{k+}$  to the current graph transforming  $G_i^k$  into  $G_i^{k+1} = G_i^k \setminus G_i^{k-} \cup G_i^{k+}$ . The associated trial move for the current level may be given by solving the MCMP on graph  $G_i^{k+1}$ , thus yielding a new feasible schedule. Due to the inherent time complexity of determining an exact solution for the MCMP at each level of the chain, the algorithm considers a *trial function* that implicitly reflects the potential quality of the trial solution that could be reached. Once the chain ends, the explicit evaluation of  $N_2$  is carried out by solving the MCMP on the graph  $G_{i+1} = G_i^{k^*}$ , where  $k^*$  represents the level of the chain where the best value of the trial function was found.

A set of six real time tables involving over 700 trips and up to 26 trains (number of blocks) is used in order to test the performance of the algorithm. The quality of the solutions is evaluated on the basis of three correlated performance measures: the percent improvements to the number of duties obtained by alternative schedulers, the matching ratio (i.e. percentage of duties with two pieces of work), and the average number of driving hours per duty. The results disclose that the ejection chain algorithm performs extremely well across the three evaluation criteria. The algorithm finds better schedules than previous methods for all problems tested, reducing the number of duties, improving the distribution of the crew's workload and finding higher matching ratios.

#### 4.4 Quadratic Assignment

The quadratic assignment problem (QAP) is a classical combinatorial optimization problem that has garnered much attention due to both its large number of applications and its solution complexity. Originally used to model a location problem in the 1950's, the QAP is computationally very difficult to solve which makes it an ideal candidate for testing new algorithmic approaches. While facility location problems remain the most popular application area for the quadratic assignment problem, many other applications for this problem exist including scheduling problems, statistical data analysis, information retrieval, as well as problems in transportation. The attractiveness of the QAP is also due to the fact that many other combinatorial optimization problems can be formulated as a QAP, including: the traveling salesman problem, the maximum clique problem and the graph partitioning problem. (See Cela (1998) for a survey of both classical and practical applications.)

In the context of facility location problems, the QAP can be stated as follows. Given a set  $F = \{f_1, \dots, f_n\}$  of  $n$  facilities to be placed in exactly  $n$  locations represented by the set  $L = \{l_1, \dots, l_n\}$ . Let  $A = (a_{ik})$  be a matrix of *distances* between pairs of locations  $l_i, l_k \in L$ , and an associated matrix  $B = (b_{jl})$  of *flows* to be transmitted (or shipped) between pairs of facilities  $f_j, f_l \in F$ . The objective is to find a minimum cost assignment of facilities to locations considering both the flow of materials between facilities and the distance between locations.

In mathematical terms, each assignment can be defined as a permutation  $p$  of the underlying index set  $N = \{1, \dots, n\}$ . Hence, if facility  $j$  is assigned to location  $i$  and facility  $l$  is assigned to location  $k$ , the cost of the flow between facilities  $j = p(i)$  and  $l = p(k)$  is  $a_{ik}b_{p(i)p(k)}$ . The QAP is the problem to find a permutation vector  $p \in P_n$  that

minimizes the total assignment cost, where  $P_n$  is the set of all possible permutations of  $N$ . Such a formulation can be generically described as

$$\text{Minimize } \sum_{p \in P_n} \sum_{i=1}^n \sum_{j=1}^n a_{ij} b_{p(i)p(j)}.$$

Heuristic approaches for the QAP abound in the literature wherein local search is commonly used as a basic component to explore the solution space. Local search methods rely on the exploration of a defined neighborhood. In the case of the QAP, this neighborhood is typically a 2-exchange neighborhood that swaps the location of two facilities at each step of the local search process. The exploration of larger neighborhoods where the simultaneous movement of  $k$  nodes of the permutation can be examined is attractive though computationally very demanding.

Ahuja et al. (2002) introduce a very large scale neighborhood search (VLSN) for the QAP, which constitutes an important advance in the creation of more complex neighborhoods for the problem. This algorithm iteratively examines all paths (or exchanges of nodes) of increasing depth, where the maximum depth is a specified parameter. The VLSN algorithm considers all moves (or a defined subset of moves) of a given depth before proceeding to the next depth. Due to the computational complexity of the full path enumeration scheme presented, a maximum path length of 4 was settled upon in their study.

More recently, Rego, James and Glover (2006) developed a specialized ejection chain algorithm for the QAP, drawing on a proposal sketched in Glover (1991), that affords additional advances. The approach utilizes the ejection chain structure to build successively larger exchanges based upon the elements chosen in the proceeding chain. In this manner, all possible chains at each depth may not be considered for a given permutation. However, this process allows the method to quickly probe larger neighborhoods, with no constraints on the depths examined, by constructing these chains of moves based upon previously promising structures.

The method may be described by analogy with the node-based ejection chain model previously discussed for the VRP. In such a model facilities are associated with nodes in a graph which are to be assigned to locations. In this context the method implements a type of *multi-node exchange move*, which can be seen as a series of swap moves for the QAP. The method begins by identifying the best local move for each facility  $j$ , which constitutes removing  $j$  from its current location and relocating it in the position occupied by a facility  $l$ , which is thereby ejected. (Alternatively, the method can start by looking at each  $l$  and finding the best  $j$  to replace it.) The initialization process is completed by simply selecting initial chains based on performing a series of best 2-exchange moves. Notably, such a move corresponds to simultaneously determining the best initial node to be ejected and the best node to occupy the location of the ejected node. The chain grows by selecting a new node to be ejected by the previously ejected node. Under the natural and convenient restriction that prevents an element from being moved twice, the chain can continue to grow until all  $n$  nodes have been ejected.

By embedding this ejection chain method within a tabu search framework, strategic control over the formation of the chains can be exerted. However, the method is applied without the benefit of advanced memory strategies, except of the simplest form, in the role of “bookkeeping” operations instead of in the role of performing advanced guidance. The objective is to show that even this very basic and unenhanced approach is

competitive with the best strategies that instead rely extensively on metaheuristic guidance to achieve their results.

Results obtained on a standard set of 22 benchmark problems from the QAPLIB demonstrate the capabilities of the raw ejection chain procedure and the average improvement obtained by exploring the larger neighborhoods over a traditional 2-exchange and previous large neighborhood approaches. Tests over 10 runs for each procedure embedded in a very simple tabu search show that the ejection chain neighborhood improved the average solution quality for 19 out of the 22 problems over its 2-exchange counterpart. Two multi-start tabu search variants are also presented, which essentially differ by the solution from which the algorithm is restarted. These enhanced variants improve the simple tabu search variant in all but 2 problem instances each, thus demonstrating the power of embedding the proposed ejection chain method within a more sophisticated local search or metaheuristic approach.

Comparisons established with two variants of the VLSN that provide the best overall quality shown that the all variants of the ejection chain algorithms significantly outperform both VLSN approaches. Specifically, the average solution quality for VLSN approaches over the 10 runs is 2.7% and 3.3% across all problems for each of the approaches, while these averages are 0.73%, 0.42%, and 0.33% for the three ejection chain methods, respectively. With respect to averages to individual problems, the simple tabu search finds better solutions than both VLSN approaches for 17 out of the 22 problems. Moreover, the best solutions obtained by the two multi-start ejection chain approaches are better than the best solutions found by the VLSN approaches in all cases.

## 5. Filter-and-Fan Applications

A filter and fan algorithm requires the definition of component moves used to generate trial solutions throughout the search process. Component moves are characteristically simple moves serving as building blocks for the construction of an extended filter and fan neighborhood. As in customary local search methods, different applications require appropriate neighborhood structures to explore the solution space. The following sections illustrate how filter-and-fan has been successfully used to create effective neighborhoods for a number of applications.

### 5.1 Facility Location

The uncapacitated facility location problem arises in bank account location planning, location of collection centers or lock-boxes, clustering analysis, location of off-shore drilling platforms, machine scheduling and information retrieval, portfolio management, and design of communication networks. For a survey see Cornuéjols, Nemhauser and Wolsey (1990) and Gao and Robinson (1994). The basic form of the problem can be defined as follows. Given a set  $S = \{1, \dots, s\}$  of warehouses or facility locations and a set  $D = \{1, \dots, d\}$  of customers to be served. With each customer  $j \in D$  is associated a demand  $b_j$  and  $c_{ij}$  is the transportation cost of completely serving a customer  $j$  by facility  $i \in S$ . Also, there is a fixed cost  $F_i$  if facility  $i$  is built (or opened). The objective is to find a set  $W^* \subseteq S$  of opened facilities that minimizes the total cost. Due to the absent of capacity constraints on the facilities, customer demands may be normalized to  $b_j = 1$  and for any set  $W \subseteq S$  of facilities there is at least one optimal assignment where all customers are served by the nearest open facility. Consequently, a UFLP solution can be fully defined by the set of open facilities. Therefore, especially in local

search, it is natural to use a vector representation  $Y = (y_1, \dots, y_s)$  where  $y_i = 1$  if the facility  $i$  is open and 0 otherwise.

Local search algorithms for the facility location problem typically use flip-based neighborhoods, namely, the *switch*-neighborhood that switches the status of one facility from open to close or vice versa by flipping a single variable at a time and the *swap*-neighborhood that simultaneously closes one facility and opens another.

Greistorfer and Rego (2006) have successfully enhanced the performance of these neighborhoods by generating sequences of flip moves within a filter-and-fan approach. Computational tests, whose outcomes are described below, disclose that this method provides a significant advance for solving facility location problems effectively. The method proceeds by performing moves that flip the value of one variable at each node of the F&F tree. A swap move implicitly results whenever in two successive nodes of a given branch of the tree, one variable flips from 0 to 1 and another variable flips from 1 to 0. The legitimacy conditions on the selection of  $\eta_2$  moves are defined by tabu restrictions preventing reverse flips (that would lead to duplicated solutions) and a feasibility condition that keeps the method from closing the only open facility in the current solution. Two variants of the algorithm are developed to achieve different levels of sophistication.

The general F&F algorithm undertakes two fundamental steps. The first step is a classical local search procedure that starts with all facilities open, then improves that solution by closing the facility that locally minimizes the objective function value and the process is repeated until no improvement is possible by closing a new facility. Let  $M$  be the set of all moves evaluated in the last iteration of this descent process, then the method keeps the  $\eta_0$  best moves of  $M$  to create the initial candidate list  $M(0)$  for the F&F tree used in the next step.

Two variants of the algorithm are implemented to achieve different levels of sophistication and performance. In a more rudimentary design, memory structures are limited to the tabu restrictions implicitly defined in the legitimacy conditions specified above. In a more advanced design, the method is enhanced by exploring *multilevel candidate lists*, which extends the legitimacy conditions with a validity check, with respect to the current depth of the search, that has its counterpart in the notion of admissibility of tabu search memories. Accordingly, the evaluation of a move may not exclusively rely on the net change in the objective function value created by the move but may include a bias factor introduced by memory considerations used to guide the search at different layers. In the present algorithm, layers are associated with two consecutive levels of the F&F tree that are subsequently and alternatively checked with respect to the solution cost changes yielded by the corresponding moves. As a result of these effects, improving moves are always kept in the tree; however if in the previous level a non-improving move was performed and if none of the moves available improve the solution at the current level, a reverse flip move that transforms the current solution back to the one in the previous level is allowed, denoting a relaxation by cost of one of the legitimacy constraints.

It is shown that the simple version is competitive with state-of-the-art algorithms, but fails to find 2 optimal solutions out of 45 classical benchmark problems. Overall this algorithm produces solutions that are on average exceedingly close to optimal, while consuming a very small amount of computation time – yielding solutions that are on average 0.04% above optimality in an average computation time of 2.78 seconds.

The more advanced version of the method was implemented with the goal of producing still better outcomes and specifically of tackling the new 60 instances currently known as the hardest UFLP data sets in literature. This version succeeded in finding all best known solutions for the previous 45 instances and achieved an average deviation of only 0.03% above the optimal solutions for the hardest 60 instances. The total time required to solve these hard problems averaged less than 3.5 seconds (on a Pentium IV, 1.7GHz CPU desktop computer).

The exceedingly high quality of these results discloses that the filter-and-fan approach provides a very effective framework to explore the solution space in facility location problems and suggests its use in other more complex variants of these problems.

## 5.2 Protein Folding

A protein's function is closely related to its 3D structure, and therefore to determine how a protein functions one must know its 3D conformation. The Protein Folding Problem (PFP) is the problem of predicting the three-dimensional (3D) structure of a protein given only the protein's sequence of amino acids. This is a fundamental yet open problem in the fields of biological chemistry and protein science, and has recently attracted attention in bioinformatics and computational biology. The PFP is central in a number of practical applications including the designing of new proteins having desirable functions in pharmaceutical, food, and agriculture industry (Lengauer 1993). We refer to Richards (1991) and Chan and Dill (1993) for an overview of the PFP and its applications.

The PFP is a notoriously difficult combinatorial problem due to the combinatorial explosion of valid conformations as the number of amino acids in the chain increases. Due to the complex nature of the PFP, the so-called HP lattice model proposed by Dill (1985) constitutes a well established simplification for algorithm assessment.

Rego, Li and Glover (2006) consider the two-dimensional (2D) version of the HP lattice model and propose a F&F algorithm for the solution of the associated PFP. A sequence of H and P amino acids is configured as a path on a two-dimensional (2D) lattice to define a valid conformation. The path designation implies that the conformation is both connected and self-avoiding, i.e., no amino acids can collide in the same cell of the lattice. (In graph theory terminology, such a path is called *node simple*.) The energy function is defined by the number of pairs of H nodes that are *adjacent* in the lattice and not *consecutive* in the chain. Each of these pairs, generally called an H-H contact, decreases the energy value by one unit. The objective is to find a conformation that minimizes the total energy of the given amino acid sequence, which therefore corresponds to maximizing the number of H-H contacts.

In this application, the F&F approach is used to seek an effective guidance strategy within a simpler neighborhood by extending the so-called *pull-move* neighborhood (Lesh, Mitzenmacher and Whitesides 2003). To elaborate the algorithm we first describe the associated *component moves* defined by the pull-move neighborhood structure.

A pull-move is initiated by moving one node of the current conformation to one of its empty diagonal adjacent positions in the square induced by the node and one of its adjacent neighbors in the sequence. Depending on the structure of the conformation the displacement of the initiating node may require other nodes to change their current positions in order to preserve connectivity. In a pull-move, displaced nodes are only allowed to occupy vacant adjacent positions in the lattice. Consequently, the preservation of connectivity also results in a self-avoiding path. Rego et al. differentiate

only three types of pull-moves designated by *filling*, *single-pull* and *multiple-pull*, according to the number of nodes that are pulled by the first displaced node. The *filling* move is the simplest pull-move, displacing a single node in the structure. A valid conformation is obtained by simply moving a node to its diagonal adjacent position. A *single-pull*, on the other hand, requires another node to change position after the initiating node takes a new position. The *multiple-pull* move extends the pull-move to achieve connectivity in more complex structures that become disconnected upon performing a single-pull move. Figure 4 shows an example of the filter and fan neighborhood for a 2D HP model with 20 amino acids, where  $\eta_1 = \eta_2 = 2$  and  $L = 3$ . The negative numbers denote the energy value of the corresponding conformation.

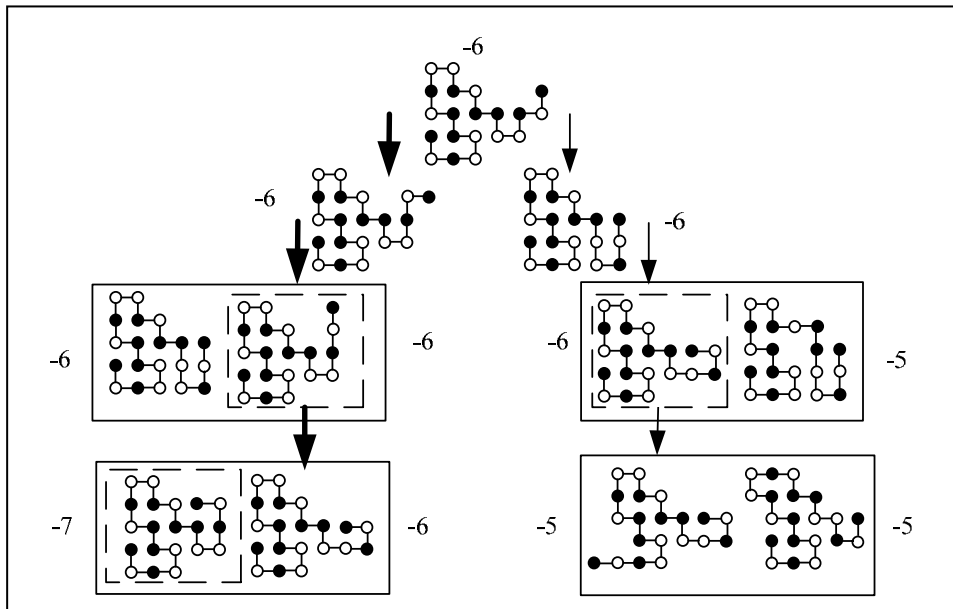


Figure 4 - Filter-and-Fan neighborhood tree for the 2D HP model of the PFP

A conformation of energy -6 (represented by the root node) denotes a local optimum determined by the local search phase. The first level of the filter and fan neighborhood is then generated by applying the  $\eta_1=2$  best moves to the root conformation. The next level is created by applying the  $\eta_2=2$  best pull-moves to each of the conformations in the current level, thus generating  $\eta_1 \cdot \eta_2=4$  trial conformations from which a new set of  $\eta_1=2$  best conformations is chosen to initiate the next level. If at one level more than  $\eta_1$  solutions exist with the same objective value preference is given to solutions that derive from different parent conformations. In the figure, the  $\eta_2$  different conformations derived from the same parent conformation are contained within the rectangles delimited by solid lines whereas the  $\eta_1$  best conformations selected at each level are contained within “interior rectangles” delimited by dotted lines. The method continues expanding the neighborhood until the improved conformation of energy -7 is found in level 3 of the filter and fan tree. The compound move leading to the improved conformation is then identified by the path indicated by the dark arrows. Note that to continue the tree search after obtaining the new local optimum, the method will restrict attention to solutions in the left-hand side branches as a basis for extending the tree.

Local search that utilizes memory of elite solutions and their attributes (either in direct or statistical form) and that strategically drives the search into new regions plays a critical role in the performance of the leading metaheuristic algorithms for the PFP.

Rego, Li and Glover (2006) explore mechanisms for achieving these aspects by proposing a filter-and-fan approach making use of a simple tabu search structure. The algorithm alternates between single-path and multiple-path tabu searches using component moves provided by the pull-move neighborhood, subject to short-term memory controls.

Computational results for a standard set of benchmark problems showed that the F&F algorithm performs more robustly and efficiently than the current leading algorithms requiring only a single solution trial and approximately 10 seconds on average to obtain best known solutions to 9 out of 11 problems. By contrast, the best of the alternative methods require a hundred or more trials in the typical case to obtain best solutions to these 9 problems. On the remaining 2 problems, F&F obtains the best known solutions after five trials (i.e. re-starts) from diverse solutions generated in the course of the algorithm, which accounted for approximately 10 hours of computation time. Noticeably, only a single run of 15 minutes on average was necessary for the F&F algorithm to find a solution that is just one unit away from the best known solution obtained by the best alternative method within similar amount of time. By comparison, this best alternative algorithm required hundreds of runs and more than 3 days (approximately 78 hours) of wall clock time just to find the best solution for one of the initial 9 instances, though the filter-and-fan algorithm requires approximately 6 seconds on an equivalent computer to find such a solution.

The success of the algorithm in performing more efficiently and robustly than alternative state-of-the-art algorithms owes to two fundamental components: (i) the dynamic and adaptive feature of the search method in exploiting the pull-move neighborhood structure; and (ii) the interplay between the tabu search and the tree search phases that creates a strategic oscillation between intensification and diversification. Further improvements in efficiency are anticipated to result by incorporating longer-term tabu search memory components to achieve higher levels of intensification, and by means of vocabulary building strategies that incorporate ejection chain methods and path-relinking.

### **5.3 Job Shop Scheduling**

The Job Shop Scheduling Problem (JSSP) is a notoriously difficult problem in combinatorial optimization. The problem finds its application in manufacturing industries and is central to many supply chains that integrate production planning and scheduling. In a supply chain environment, production planning and scheduling models are often incorporated into a unified framework, sharing information and interacting with one another in order to optimize the production of different products over multiple facilities. The output of the planning process serves as an input to the scheduling process, which is often analyzed as a job shop scheduling problem. Planning and scheduling models may also interact with other types of logistics models such as inventory models, facility location models and transportation models. For an extensive coverage of planning and scheduling models and applications in various supply chains settings, see Pinedo (2006).

The JSSP can be defined by a set of machines specialized to perform ordered operations unique for every job. No machine can perform more than one operation at a time, each operation has fixed time duration, and preemption is not allowed. The goal is to minimize the duration of the longest job in the schedule (i.e. the makespan).

Beam Search is a classical tree search method typically used in the optimization of complex scheduling systems, including the JSSP (Sabuncuoglu and Bayiz, 1999);

however more advanced forms of tree search neighborhood approaches have been recently proposed and successfully applied to scheduling as well as to several other optimization problems.

In particular, Balas and Vazacopoulos (1998) consider a specialized neighborhood tree for the JSSP that lead to one of the most effective algorithms for this problem. Making use of this neighborhood, Rego and Duarte (2006) developed a filter-and-fan (F&F), which can be viewed as a natural generalization of *beam search* and which includes the B&V neighborhood tree as a special case. The basic structure of the algorithm can be described as follows.

The most rudimentary version of the classical *shifting bottleneck procedure* (SBP) (Adams, Balas and Zawack, 1988) is used as a constructive method to generate an initial feasible solution. At each step the machine with longest processing time (i.e. the bottleneck machine) among the ones that have not been scheduled is selected for scheduling and the method stops when all machines are scheduled. It is well-known that this procedure does not produce high quality solutions by itself, but provides a convenient means to rapidly generate initial feasible solutions for more advanced algorithms.

The F&F algorithm starts from the solution generated by the SBP and iteratively improves this solution by alternating between the local search and the tree search phases. The method considers two types of neighborhoods  $N_1$  (Aarts et al., 1994) and  $N_2$  (Nowichi and Smutnicki, 1996) based on classical moves that swap two adjacent operations in the critical path (i.e. the longest path in the problem graph that represents the solution). Typically,  $N_1$  swaps arcs that are internal to the blocks of operations in the same machine while  $N_2$  exploits interactions between adjacent blocks by swapping arcs linking operations in different blocks. Depending on the search strategy both types of moves can be used for the local search as well as to define elementary moves in the F&F tree.

The search starts with the  $N_1$  neighborhood. Any time a local optimum is found (in the local search phase) the best  $M(0)$  moves (among the  $M$  moves evaluated to establish local optimality) are used to create the first level of the F&F neighborhood tree. The next levels are created using  $\eta_1=16$  and  $\eta_2=8$ . The method stops branching as soon as an improved solution is found, the maximum number of levels  $L$  is reached, or if there is no more legitimate candidate moves to evaluate.

In case a global improvement is found in the tree search the new best solution is made the starting solution for another run of the local search procedure. However, if the solution at the root node can not be improved, the method switches back to the local search starting with the best trial solution encountered in the tree search and using neighborhood  $N_2$ . In this case, the list  $M$  determined in the last run of the local search procedure, and so made up of type  $N_1$  moves, is now extended with new candidates of type  $N_2$ . The new list  $M(0)$  is created using the best moves of each type in equal number. The objective is to allow the algorithm to combine both types of neighborhoods throughout the F&F tree.

The performance of the algorithm was evaluated on a set of 58 benchmark problems belonging to four classical sets known as LA, FT, ABZ, and ORB.

The analysis of the computational results shows that the F&F algorithm produces solutions that are on average at 0.80% above the optimum (or best known) solutions for over all problem instances. The algorithm is also very fast, finding its best solutions in relatively short time (on a 1.7 GHz Pentium IV 256MB): less than 40 seconds on average



for ABZ problems and no more than 8 seconds on average for classes FT, LA and ORB. Also, only 1 second of running time was enough for the algorithm to find the optimal solution for 30 out of the 38 instances that the method successfully solved to optimality.

We have also compared our approach to two leading methods found to be the best among thirteen methods tested in a recent study by Gonçalves, Mendes and Resende (2005): a hybrid genetic algorithm/local search (GA/LS) method developed as part of the study, and a tabu search (TS) approach by Nowichi and Smutnicki (1996), which emerged the clear winner of all methods examined. The performance of the present F&F approach with regard to solution quality places it next after the TS approach, with an average relative deviation from the best known solutions of 0.33%, as compared to 0.05% for the TS approach and 0.39% for the hybrid GA/LS approach. The F&F approach also falls between these two other methods in solution speed, running about an order of magnitude slower than the TS approach, but about 2 orders of magnitude faster than the GA/LS approach (after adjusting for differences in computers). However, the F&F procedure emerges as significantly more robust than the other two methods in the time required to find best solutions. F&F times range from 1 to 44 seconds with a standard deviation of 9.7, while the TS times range from less than 1 second to 623 seconds with a standard deviation of 147.6, and the GA/LS times range from 13 to 3745 seconds with a standard deviation of 1183.0. However, another TS algorithm for the JSSP has recently emerged that appears to be substantially better yet in relation to both speed and robustness. The tabu search approach due to Grabowski and Wodecki (2005), finds solutions for the same testbed discussed having an average relative deviation from the best known of 0.08% in about 1.09 seconds (on a 333 MHz CPU), a time that would be insignificant if runs were performed on a faster computer like the ones used by the F&F and the GA/LS algorithms.

#### 5.4 Capacitated Minimum Spanning Tree

The capacitated minimum spanning tree problem (CMST) has been addressed extensively in the literature for its importance in modeling and practical applications. It is fundamental to the design of communication networks and encounters its application in a variety of other settings chiefly in the areas of distribution, transportation and logistics. For background on applications we refer the reader to Gavish (1982, 1991). From the modeling standpoint, the problem appears as a relaxation of the classical capacitated vehicle routing problem, which in turn is central in many other more complex problems. See Amberg, Domschke and Voß (1996) and Mathew and Rego (2006) for a comprehensive review of methods and solution approaches.

The CMST problem can be stated as follows. Given a complete undirected graph  $G = (V_0, A)$ , where  $V_0 = \{v_0, v_1, \dots, v_n\}$  is a vertex (node) set and  $A = \{(v_i, v_j) | v_i, v_j \in V; i \neq j; j \neq 0\}$  is an arc set. Let  $v_0$  denote a special *central* node (root), and let  $V = V_0 \setminus \{v_0\}$  be a set of *terminal* nodes requiring a specified demand  $d_i$ .  $C = (c_{ij})$  is an  $n \times n$  matrix associated with  $A$ , where  $c_{ij}$  is a non-negative weight (distance or cost) on arc  $(v_i, v_j)$  if there is an arc between  $v_i$  and  $v_j$ . Otherwise  $c_{ij}$  is infinity. The CMST problem consists of finding a minimum cost tree  $T$  spanning all nodes of  $G$ , so that the sum of the demands in each sub-tree incident to the root node does not exceed a fixed arc capacity  $Q$ . When all the nodes  $v_i \in V$  have the same demand the problem is referred to as the homogenous demand CMST problem.

Successful approaches to the CMST problem involve high complexity multi-exchange neighborhoods that take advantage of the basic tree-based and node-based neighborhoods used in tabu search algorithms to address the problem. Node-based neighborhoods generate moves that transfer a node from one sub-tree to another or exchange nodes between sub-trees, while tree-based neighborhoods transfer sub-trees between different sub-trees.

The evaluation of node-based or tree-based neighborhood in dense graphs requires  $O(n^2)$  effort, and the effort to evaluate a combination of  $L$  of these moves is  $O(n^L)$ , and hence grows exponentially with  $L$ . A potentially best combination of  $L$  moves can be evaluated with significantly less effort if the combination is thought of as a compound move consisting of individual moves evaluated progressively using the filter-and-fan strategy.

The effectiveness of the filter-and-fan method for implementing complex compound moves that improve the local optima with only a modest increase in computational effort is examined in Mathew, Rego and Glover (2006). This algorithm uses a design of the F&F approach wherein the descent phase is replaced with a tabu search phase and the tree search is continued after a local optimum is found, allowing for local optima to be overcome in any level of the tree except for leaf nodes. In addition the method employs a neighborhood structure that brings about two types of strategic oscillation: (1) cycling between feasibility and infeasibility and (2) cycling between node-based shift moves and tree-based shift moves. Strategic oscillation is a specialized tabu search technique that operates by orienting the search with respect to some boundary. In a one-sided oscillation, which is appropriate for the present setting, whenever such a boundary is reached the algorithm changes direction according to a specified search mechanism. In this algorithm changing direction involves switching to the alternate neighborhood structure. The memory structures used include short term memory defined by the classical tabu restrictions and aspiration criteria together with critical event memory to bring about strategic oscillation. A brief description of the algorithm follows.

A complete evaluation of both the node-based and tree-based neighborhood is performed incorporating penalty costs for moves that lead to infeasible solutions on the initial solution  $X_0$  and a set of  $\eta_0$  best moves that lead to solutions with the lowest objective function values is selected. From among these moves a subset of the  $\eta_1$  best moves (which can be either node-based or tree-based) are executed to form  $\eta_1$  different solutions. For each of these solutions,  $\eta_2$  highest evaluating moves from the original  $\eta_0$  are selected. From the union of the  $\eta_2$  moves for all  $\eta_1$  solutions, the best  $\eta_1$  moves are executed to produce  $\eta_1$  new solutions. This process extends for  $L$  levels of the F&F tree (in a diversification phase) or until the best solution is improved upon (in an intensification phase), in which case the process is repeated from the beginning using the best solution encountered throughout the tree to re-initiate the tabu search phase. In this manner the filter-and-fan approach brings about simple tree-based and node-based shift moves that consist of at most  $L$  moves. The resulting compound move avoids the computational overhead required for the complete neighborhood evaluation necessary to determine the exact best  $L$ -compound move. Additionally, to prevent cycling in the solution space, the most recent moves executed are maintained as tabu active for a stipulated number of iterations.

Computational tests performed using standard benchmark problems revealed that this algorithm produced results that compared favorably to a number of prior metaheuristic algorithms and rivals the best. Tests were carried on a total of 125 instances comprising 45 heterogeneous demand problems and 80 homogenous demand problems. The algorithm managed to find the best known solutions in 70 of these 125 instances with

an overall deviation of 0.65% on average. In addition, the average execution time for the F&F approach proved to be significantly lower than that of the state-of-the-art competitors on comparable platforms.

For an appropriate comparative analysis, runs were performed in a similar manner and on the same groups of instances considered by the alternative algorithms. Amberg, Domschke and Voß (1996) tested a simulated annealing algorithm and six variants of a tabu search algorithm on 70 homogeneous-demand instances. The various algorithms differ by the type of neighborhood and the method used to manage the tabu restrictions in the tabu search algorithms. Results obtained for 12 independent runs of each algorithm disclosed that a run of the F&F algorithm is better than any of the runs of these algorithms. In particular, it is shown that even if all seven variants of these algorithms are taken together and the best overall run for each individual problem is chosen, the average quality of the solutions produced by the F&F algorithm across all problems improves the quality of such best solutions by 0.43%, indicating a clear dominance of the F&F algorithm over these alternative strategies for the CMST. Similar analysis reveals a significant advantage of the F&F algorithm over the tabu search implementation of Shariha et al. (1997) and the adaptive reasoning technique (ART) of Patterson, Pirkul and Rolland (1999) across all problem categories. More competitive approaches are due to Ahuja, Orlin and Sharma (2001) who propose two very large-scale neighborhood search (VLSN) approaches based on multi-exchanges of node-based and tree-based neighborhoods, respectively. These neighborhoods are used to create two different variants of a tabu search and a GRASP algorithm. Tests on 2 groups of problems of different sizes and characteristics indicate that the F&F algorithm performs better than one of the GRASP variants for the first group and better than the other GRASP variant for the other group. Similarly, the F&F algorithm performs better than one of the TS variants for a group of problems and is very competitive with the other variant for the other group. As an overall assessment, the F&F approach outperforms a GRASP and a TS variant. A considerable advantage of the F&F algorithm concerns the significantly reduced amount of solution time required by this method relative to the solution times required by each of the 4 variants of the competing algorithms.

These results clearly indicate the impact of the neighborhood structure in the performance of metaheuristic strategies. In particular, node-based neighborhoods prove more appropriate for solving homogeneous-demand problems while tree-based neighborhoods have particular advantages for solving heterogeneous-demand problems. To take advantage of the complementary features of the two types of neighborhoods, a strategy that unifies node-based and tree-based into a composite multi-exchange neighborhood has been proposed in Ahuja, Orlin and Sharma (2003) to produce an enhanced GRASP implementation. The neighborhood search is powered by an exact dynamic programming solution method aimed at finding the best move in the composite neighborhood. This enhanced variant finds all best known solutions for the 75 problems tested (out of the 125 considered by the F&F algorithm), and so proves relatively more effective, although to achieve this result the method requires more than four times as much effort as the F&F method (on a similar computer) to find solutions of the same or exceedingly similar quality.

## **6. Conclusion**

Important advances in local search have resulted from the development of larger neighborhoods, organized in structurally exploitable ways that are capable of exploring the solution space more extensively at each iteration. Such neighborhoods allow for a broader examination of the solution landscape and yield more choices to perform moves that offer the potential to find regions of high quality solutions. Advances in this domain have particularly arisen from compound neighborhood structures, which combine

simple neighborhoods to create more complex neighborhoods that can be explored to variable depths. To take advantage of the potential to find better solutions, however, careful attention must be given to managing the computational overhead involved in generating and searching compound neighborhoods, due to the greater number of operations required to process them by comparison to the simpler neighborhoods of which they are composed. Accordingly, a number of studies have investigated strategies to combine neighborhoods efficiently, and thereby reduce the computational effort of generating solution trajectories that they make available.

In this paper, we focus on *ejection chains* and *filter-and-fan* methods, which have become the source of significant advances in the construction of very large neighborhood structures. In addition to presenting the general framework of these methods, we elucidate the key considerations underlying their design and successful implementation. We further identify specific ejection chain and filter-and-fan algorithms that have proved effective in the solution of problems spanning the domains of facility location, routing and distribution, production scheduling, network design, resource allocation, manpower planning, and computational biology. By this means, we undertake to provide insights that may prove useful for developing more effective algorithms in a variety of additional settings.

Finally, we briefly comment on issues that are relevant for determining whether a filter-and-fan approach may be preferable to an ejection chain approach, or vice versa. Evidently, the merit of applying one method or the other depends on the application, the complexity of the problem and ultimately on the search strategy embodied in the adaptive memory process. As a rule of thumb, in settings where simple neighborhoods have proved relatively effective (or very effective for instances of size relatively smaller than the ones at hand), methods that rely on these simple neighborhoods can very likely be enhanced for more challenging applications by a filter-and-fan approach. Conversely, in complex applications where classical neighborhoods are rather limited in their ability to explore the solution space, particularly in the case of very large problem instances, a method based on an ejection chain design is likely to prove of greater value. While ejection chain approaches are characteristically more powerful than filter-and-fan approaches, they are usually more difficult to implement and less flexible for being modified to handle changed problem specifications. Since advanced ejection chain methods typically involve relatively complex reference structures, they are usually more difficult to adapt to handle new requirements and constraints. In those applications where requirements are likely to change over time, the question of the preferred method to use thus depends on the tradeoff between the value of obtaining the best possible solution and the value of being able to adapt the method to meet new conditions with a modest outlay of effort.

## References

Aarts, E.H.L., P.J.M. Van Laarhoven, J.K. Lenstra and N.L.J. Ulder (1994) "A Computational Study of Local Search Algorithms for Job Shop Scheduling," *ORSA Journal on Computing*, 6(2), 118-125.

Adams, J., E. Balas and D. Zawack (1988) "The Shifting Bottleneck Procedure for Job Shop Scheduling," *Management Science*, 34 (3), 391-401.

Ahuja, R.K., J.B. Orlin and D. Sharma (2002) "Very Large-Scale Neighborhood Search for the Quadratic Assignment Problem," submitted to *INFORMS Journal on Computing*.

- Ahuja, R.K., J.B. Orlin and D. Sharma (2001) "Multi-Exchange Neighborhood Search Structures for the Capacitated Minimum Spanning Tree Problem," *Mathematical Programming*, 91, 71-97.
- Ahuja, R.K., J.B. Orlin and D. Sharma (2003) "A Composite Very Large-Scale Neighborhood Structure for the Capacitated Minimum Spanning Tree Problem," *Operations Research Letters*, 31, 185-194.
- Amberg, A., W. Domschke and S. Voß (1996) "Capacitated Minimum Spanning Trees: Algorithms using Intelligent Search," *Combinatorial Optimization: Theory and Practice 1*, 9-39.
- Applegate, D., W. Cook and A. Rohe (2003) "Chained Lin-Kernighan for Large Traveling Salesman Problems," *INFORMS Journal on Computing*, 15, 82-92.
- Balas, E. and A. Vazacopoulos (1998) "Guided Local Search with Shifting Bottleneck for Job Shop Scheduling," *Management Science*, 44 (2), 262-275.
- Cavique, L., C. Rego and I. Themido (1999) "Subgraph Ejection Chains and Tabu Search for the Crew Scheduling Problem," *Journal of Operational Research Society*, 50, 608-616.
- Cao, B. and F. Glover (1997) "Tabu Search and Ejection Chains: Application to a Node Weighted Version of the Cardinality-Constrained TSP," *Management Science*, 43(7), 908-921.
- Cela, E. (1998) "The Quadratic Assignment Problem: Theory and Algorithms," Kluwer Academic Publishers, Boston.
- Cirasella, J., D.S. Johnson, L.A. McGeoch and W. Zhang (2001) "The Asymmetric Traveling Salesman Problem: Algorithms, Instance Generators and Tests," in *Proceedings of the Algorithm Engineering and Experimentation, Third International Workshop, ALENEX 2001*, 32-59.
- Chan, H.S. and K.A. Dill (1993) "The Protein Folding Problem," *Physics Today* 46(2): 24-32.
- Chrobak, M., T. Szymacha and A. Krawczyk (1990) "A Data Structure Useful for Finding Hamiltonian Cycles," *Theoretical Computer Science*, 71, 419-424.
- Cirasella, J., D.S. Johnson, L.A. McGeoch and W. Zhang (2001) "The Asymmetric Traveling Salesman Problem: Algorithms, Instance Generators and Tests," in *Proceedings of the Algorithm Engineering and Experimentation, Third International Workshop, ALENEX 2001*, 32-59.
- Cornuéjols, G., G.H. Nemhauser and L. Wolsey (1990) "The Uncapacitated Facility Location Problem," In P. Mirchandani and R. Francis (editors), *Discrete Location Theory*, 119-171, John Wiley and Sons, Inc., New York.
- Christofides, N., A. Mingozzi and P. Toth (1979) "The Vehicle Routing Problem," In A. Mingozzi, P. Toth, C. Sandi (editors), *Combinatorial Optimisation*, 315-338, Wiley Chichester.
- Dill, K.A. (1985) "Theory for the Folding and Stability of Globular Proteins," *Biochemistry* 24(6): 1501-1509.

- Dorndorf, U. and E. Pesch (1994) "Fast Clustering Algorithms," *ORSA Journal on Computing*, 6, 141-153.
- Fisher, M.L. (1994) "Optimal Solution of Vehicle Routing Problems using Minimum k-Trees", *Operations Research*, 42, 4, 626-642.
- Fredman, M.L., D.S. Johnson, L.A. McGeoch and G. Ostheimer (1995) "Data Structures for Traveling Salesmen," *Journal of Algorithms*, 18, 432-479.
- Funke, B., T. Grünert, S. Irnich (2005) "A Note on Single Alternating Cycle Neighborhoods for the TSP," *Journal of Heuristics*, 11, 135-146.
- Gamboa, D., C. Rego and F. Glover (2005) "Data Structures and Ejection Chains for Solving Large-Scale Traveling Salesman Problems," *European Journal of Operational Research*, 160, 154-171.
- Gamboa, D., C. Rego and F. Glover (2006) "Implementation Analysis of Efficient Heuristic Algorithms for the Traveling Salesman Problem," *Computers and Operations Research*, 33, 1161-1179.
- Gamboa, D., C. Rego, F. Glover and C. Osterman (2006) "An Experimental Evaluation of Ejection Chain Algorithms for the Traveling Salesman Problem," School of Business Administration, University of Mississippi, MS.
- Gao, L.L. and E.P. Robinson (1994) "Uncapacitated Facility Location: General Solution Procedures and Computational Experience," *European Journal of Operational Research*, 76, 410-427.
- Gavish, B. (1982) "Topological Design of Centralized Computer Networks: Formulations and Algorithms," *Networks* 12, 355-377.
- Gavish, B. (1991) "Topological Design Telecommunications Networks – Local Access Design Methods," *Annals of Operations Research*, 33, 17-71.
- Glover, F. (1992) "New Ejection Chain and Alternating Path Methods for Traveling Salesman Problems," *Computer Science and Operations Research*, 449-509.
- Glover, F. (1991) "Multilevel Tabu Search and Embedded Search Neighborhoods for the Traveling Salesman Problem," Leeds School of Business, University of Colorado, Boulder, CO.
- Glover, F. (1996) "Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems," *Discrete Applied Mathematics*, 65, 223-253.
- Glover, F. (1998) "A Template for Scatter Search and Path Relinking," In J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers (editors), *Artificial Evolution. Lecture Notes in Computer Science*, 1363, 3-51. Springer, Heidelberg.
- Glover, F. and M. Laguna (1997) "Tabu Search," *Kluwer Academic Publishers*, Boston.
- Gonçalves, J.F., J.J.M. Mendes and M.G.C. Resende (2005) "A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem," *European Journal of Operational Research*, 167, 77-95.

- Grabowski, J. and M. Wodecki (2005) "A Very Fast Tabu Search Algorithm for Job Shop Problem," In C. Rego and B. Alidaee (editors), *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*, 191-211, *Kluwer Academic Publishers*, Boston.
- Greistorfer, P. and C. Rego (2006) "A Simple Filter-and-Fan Approach to the Facility Location Problem," *Computers and Operations Research*, 33 (9), 2590–2601.
- Helsgaun, K. (2000) "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic", *European Journal of Operational Research*, 126, 106-130.
- Johnson, D.S. and L.A. McGeoch (1997) "The Traveling Salesman Problem: A Case Study in Local Optimization", In *Local Search in Combinatorial Optimization*, E.H.L. Aarts and J.K. Lenstra (editors), John Wiley and Sons, Ltd., 215-310.
- Johnson, D.S., L.A. McGeoch, F. Glover and C. Rego (2000) "8th DIMACS Implementation Challenge: The Traveling Salesman Problem," <http://www.research.att.com/~dsj/chtsp/>.
- Kanellakis, P.C. and C.H. Papadimitriou (1980) "Local Search for the Asymmetric Traveling Salesman Problem," *Operations Research*, 28, 1086–1099.
- Lengauer, T. (1993) "Algorithmic research problems in molecular bioinformatics," *Proceedings of the Second Israel Symposium on Theory of Computing Systems*, ISTCS 1993, Natanya, Israel, 177-192.
- Lesh, N., M. Mitzenmacher and S. Whitesides (2003) "A Complete and Effective Move Set for Simple Protein Folding," In *Proceedings of the 7th Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, ACM Press, NY, 188-195.
- Lin, S. and B. Kernighan (1973) "An Effective Heuristic Algorithm for the Traveling Salesman Problem," *Operations Research*, 21, 498-516.
- Mathew, F. and C. Rego (2006) "Recent Advances in Heuristic Algorithms for the Capacitated Minimum Spanning Tree Problem," In *Proceedings of the 37th Annual Meeting of Decision Sciences Institute (DSI)*, 31021-31026.
- Mathew, F., C. Rego and F. Glover (2006) "A Filter-and-Fan Algorithm for the Capacitated Minimum Spanning Tree," School of Business Administration, University of Mississippi, MS.
- Nowichi, E. and C. Smutnicki (1996) "A Fast Taboo Search Algorithm for the Job Shop Problem," *Management Science*, 42(6), 797-813.
- Osterman, C. and C. Rego (2003) "The Satellite List and New Data Structures for Symmetric Traveling Salesman Problems," School of Business Administration, University of Mississippi, MS.
- Patterson, R., H. Pirkul and E. Rolland (1999) "Memory Adaptive Reasoning for Solving the Capacitated Minimum Spanning Tree Problem," *Journal of Heuristics*, 5, 159-180.
- Pinedo, M.L. (2006) "Planning and Scheduling in Manufacturing and Services," Series in Operations Research and Financial Engineering, Springer-Verlag.

- Rego, C. (1998a) "Relaxed Tours and Path Ejections for the Traveling Salesman Problem," *European Journal of Operational Research*, 106, 522-538.
- Rego, C. (1998b) A Subpath Ejection Method for the Vehicle Routing Problem. *Management Science*, 44:10, 1447-1459.
- Rego, C. (2001) "Node Ejection Chains for the Vehicle Routing Problem: Sequential and Parallel Algorithms", *Parallel Computing*, 27, 201-222.
- Rego, C. and R. Duarte (2006) "A Filter and Fan Approach for the Job Shop Scheduling Problem," School of Business Administration, University of Mississippi, MS.
- Rego, C. and F. Glover (2002). "Local search and metaheuristics for the traveling salesman problem," In G. Gutin and A. Punnen (editors), *The Traveling Salesman Problem and its Variations*, 309-368. *Kluwer Academic Publishers*, Boston.
- Rego, C., F. Glover, D. Gamboa and C. Osterman (2006) "A Doubly-Rooted Stem-and-Cycle Ejection Chain Algorithm for Asymmetric Traveling Salesman Problems", School of Business Administration, University of Mississippi, MS.
- Rego, C., T. James and F. Glover (2006) "An Ejection Chain Algorithm for the Quadratic Assignment Problem," School of Business Administration, University of Mississippi, MS.
- Rego, C., H. Li and F. Glover (2006) "A Filter-and-Fan Approach to the 2D HP Model of the Protein Folding Problem," School of Business Administration, University of Mississippi, MS.
- Richards, F.M. (1991). "The Protein Folding Problem," *Scientific American* 264(1): 54-7, 60-3.
- Rochat, Y. and E. Taillard (1995) "Probabilistic Intensification and Diversification in Local Search for Vehicle Routing", *Journal of Heuristics*, 1, 147-167.
- Sabuncuoglu, I. and M. Bayiz (1999) "Job Shop Scheduling with Beam Search," *European Journal of Operational Research*, 118, 390-412.
- Sharaiha, Y.M., M. Gendreau, G. Laporte and I.H. Osman (1997) "A Tabu Search Algorithm for the Capacitated Shortest Spanning Tree Problem," *Networks*, 29, 161-171.
- Taillard, E. (1993) "Parallel Iterative Search Methods for Vehicle Routing Problems," *Networks*, 23, 661-673.
- Yagiura, M., T. Ibaraki and F. Glover (2004) "An Ejection Chain Approach for the Generalized Assignment Problem," *INFORMS Journal on Computing*, 16, 133-151.