ELSEVIER

Discrete Applied Mathematics 76 (1997) 165–181

DISCRETE APPLIED MATHEMATICS

TSP ejection chains

Erwin Pesch[a], Fred Glover[b,*]

[a] *Institute of Economics and Business Administration, BWL 3, University Bonn, Adenauerallee 24-42, D-53113 Bonn, Germany*

[b] *US West Chair in System Science, Graduate School of Business, University of Colorado, Campus Box 419, Boulder, CO 80309-0419, USA*

Received 3 February 1995; revised 3 October 1995

Abstract

We identify ejection chain methods for the traveling salesman problem based on a special reference structure for generating constructions related to alternating paths. Computational tests show that the method performs very effectively, obtaining generally better solutions than improved versions of the Lin–Kernighan method within the same time frame. Our approach, which currently has a simple tabu search guidance component at a local level, also has the potential to be combined in more advanced ways with metaheuristics such as genetic algorithms, simulated annealing and tabu search.

1. Introduction: Variable depth methods and ejection chains

Variable depth methods, whose terminology was popularized by Papadimitriou and Steiglitz [15], have had an important role in heuristic procedures for optimization problems. The origins of such methods go back to prototypes in network and graph theory methods of the 1950s and 1960s. A class of these procedures called *ejection chain methods* has proved highly effective in a variety of applications, including clustering [3] generalized assignment (see [11]; we also refer to [9]), and vehicle routing [19, 21, 17].

Ejection chain methods extend ideas exemplified by certain types of shortest path and alternating path constructions. The basic moves for transitioning from one solution to another are *compound moves* composed of a sequence of paired steps. The first component of each paired step in an ejection chain approach introduces a change that creates a *dislocation* (i.e., an inducement for further change), while the second component creates a change designed to *restore* the system. The dislocation of the

* Corresponding author. E-mail: glover_f@cubldr.colorado.edu.

first component may involve a form of infeasibility, or may be heuristically defined to create conditions that can be usefully exploited by the second component. Typically, the restoration of the second component may not be complete, and hence, in general, it is necessary to link the paired steps into a chain that ultimately achieves a desired outcome.

1.1. Ejection chains in graphs

The *ejection* terminology comes from the typical graph theory setting where each of the paired steps begins by introducing an element (such as a node, edge or path) that disrupts the graph's preferred structure, and then is followed by ejecting a corresponding element, in a way that recovers a critical portion of the structure. A chain of such steps is controlled to assure that the preferred structure eventually will be fully recovered (and preferably, fully recovered at various intermediate stages by means of trial solutions). The candidate element to be ejected in such instances may not be unique, but normally comes from a limited set of alternatives.

The alternating path construction [2] gives a simple illustration. Here, the preferred graph structure requires a degree constraint to be satisfied at each node (bounding the number of edges allowed to meet the node). The first component of a paired step introduces an edge that violates such a degree constraint, causing too many edges to meet a particular node, and thus is followed by a second component that ejects one of the current edges at the node so that the indicated constraint may again be satisfied. The restoration may be incomplete, since the ejected edge may leave another node with too few edges, and thus the chain is induced to continue. As we will see, a construction called a *reference structure* becomes highly useful for controlling such a process, in order to restore imbalances at each step by means of special trial solution moves.

Following the broader perspective previously described, ejection chain processes of course are not limited to graph constructions. For example, they can be based on successively triggered changes in values of variables, as illustrated by a linked sequence of zero–one exchanges in multiple choice integer programming applications or by linked "bound escalations" in more general integer programs. Additional illustrations are provided in [7].

1.2. Current focus

In this paper we introduce a special type of ejection chain method for the traveling salesman problem (TSP). Our approach is based on using a *stem-and-cycle* reference structure (although we will also indicate simple adaptations to handle other reference structures). Such a construction produces a chain consisting of a generalized type of alternating path, where some edges of the path may repeat or cancel others. Legitimate tours remain accessible to the construction at each step. The approach is easy to implement, and can be embedded into metaheuristic procedures such as genetic algorithms, simulated annealing and tabu search. Without taking advantage of metaheuristics, the method is nevertheless highly effective compared to other traveling salesman approaches that represent the state of the art, as shown by our experimental tests.

2. Formulation and links to classical constructions

Our following development assumes that the reader is familiar with standard graph theory terminology. We are concerned with the symmetric TSP on a graph $G=(N,E)$, where $N=\{1,\ldots n\}$ is the set of nodes (or "cities") and E is the set of edges consisting of unordered pairs of nodes. The cost or length of an edge (i,j) (i.e., the node pair $\{i,j\}$) is denoted $c_{ij}=c_{ji}$ and our goal is to obtain a tour (Hamiltonian cycle) T of minimum cost (or length) in G, i.e., we seek a minimum value for the objective function

$$c(T)=\sum(c_{ij}\colon (i,j) \text{ belongs to } T),$$

where T ranges over the set of all possible tours. Hundreds of exact and heuristic solution procedures have been proposed for the TSP; the popular and successful ones are described in the books by Lawler et al. [12], Reinelt [18], Pesch [16] and the survey paper on the most powerful exact approaches by Applegate et al. [1].

To simplify our discussion, we suppose the graph G is complete (hence E consists of all unordered node pairs) and we allow "missing edges" to be included with infinite costs. It will be evident, however, that our observations also apply to sparse graphs, by a policy that introduces infinite cost edges only in special cases where they may be needed.

2.1. Basic connections

The TSP can be expressed as a 0–1 integer programming (IP) problem in a variety of ways. A common formulation introduces a variable x_{ij} for edge (i,j), where x_{ij} takes the value 1 if edge (i,j) belongs to the tour and takes the value 0 otherwise. Constraints are then introduced to assure the tour structure will be satisfied, among them the simple constraint

$$\sum(x_{ij}\colon (i,j)\in E)=n,$$

which assures the tour will have the proper number of edges.

In any such 0–1 IP problem where a fixed number (n) of variables must be assigned the value 1, the set of all moves for transforming one feasible solution into another is precisely the set that reverses the 0–1 assignment of some variables currently equal 1 and of an equal number of variables currently equal 0 (subject to maintaining other constraints satisfied). If r denotes the number of variables of each type, so that $2r$ variables change their values, then r may range from 1 to n. In the special case of the TSP, such moves are called *r-exchanges*, and correspond to dropping r edges from the current tour and replacing them by another set of r edges. For the TSP, to assure the edges dropped and added result in a new tour, the value r must be at least 2.

2.2. *Alternating paths and cycles*

It is easy to show that every TSP r-exchange must decompose into a collection of one or more alternating cycles (due to the fact that each node must be met by exactly two edges of the TSP tour). Consequently, this gives a natural motivation for developing ejection chain strategies based upon alternating path concepts. However, not all alternating cycles will successfully transform a given tour into a new tour, since the degree constraints can also be satisfied by any collection of subtours (a classical difficulty also encountered by linear programming relaxations of TSP formulations), and alternating cycles can generate all such collections of subtours as well. Again, the use of a reference structure is crucial for overcoming this difficulty.

The minimal 2-exchange for TSPs, often also called a "2-opt" move, corresponds to a smallest alternating cycle that can be created by reference to a current tour T. This follows from the standard graph theory definition, where an alternating path, relative to a given set T, constitutes a path in G whose even edges belong to T and whose odd edges belong to $E - T$. (Note we adopt the convention that allows T to be treated notationally as a set of edges as well as the tour containing those edges.) Consequently, since a single edge connects a given pair of nodes in G, an alternating cycle must contain at least two edges of E and two edges of $E - T$.

It is well known that a 2-exchange constitutes one of two ways of dropping two non-adjacent edges of a tour and adding two other (non-tour) edges that meet the endpoints of the dropped edges. We observe that both of these alternatives in fact correspond to (minimum) alternating cycles, and consequently this provides an elementary example of the fact that not all alternating cycles succeed in transforming a tour into a new tour.

2.3. *Basic strategy*

Our primary ejection chain approach is designed to generate moves composed of alternating cycles and associated *dynamic* alternating cycles. Our sequences of paired steps therefore introduce and eject an edge at each move. We may call these basic paired steps 1-*exchanges*, by analogy with the TSP terminology. However, in contrast with the TSP r-exchange, a 1-exchange does not transform a tour into a new tour by itself, and hence our terminology departs slightly from convention.

To obtain benefit from linking these elementary moves into a chain we must identify a way to control the process so that trial solution moves are available to recover feasible solutions at critical junctures, even though the current ejection chain construction itself may not provide such a solution. We give a way to meet this challenge in the next section.

3. The stem-and-cycle reference structure

The backbone of our construction, called the *stem-and-cycle* reference structure, may be characterized as follows [6].

The stem-and-cycle reference structure is a spanning subgraph that consists of a cycle attached to a path, called the stem. The node that represents the intersection of the stem and the cycle is called the root node, denoted by s, and the two nodes of the cycle adjacent to the root are called the subroots. The other end of the stem is called the tip of the stem, denoted by k. An example of the stem-and-cycle structure is shown in Fig. 1(a) (where $s=1$ and $k=3$). The labels of the nodes indicate the sequence in which the nodes are visited in order to modify the reference structure (starting from an initial tour). Hence, edge $(1,2)$ has been added and $(2,3)$ has been dropped from the initial tour. Node 1 is adjacent to two subroots, node 2 and the non-labeled node of Fig. 1(b).

The stem can be degenerate, consisting of a single node, in which case $s=k$ and the stem-and-cycle structure corresponds to a tour. Two trial solutions are available for creating a tour when the stem is non-degenerate, each obtained by adding an edge (k,s') from the tip to one of the subroots s', and deleting the edge (s,s') between this subroot and the root. (When the stem is degenerate, this operation adds and deletes the same edge, leaving the tour unaffected). Consider Fig. 1(b); there is one possible trial solution adding an edge connecting node 3 to the non-labeled subroot that belongs to the cycle. (In all diagrams, steps to obtain a trial solution are indicated through a dotted edge to a subroot. This subroot has no label and is adjacent to node 1.) Then, the edge yet connecting this subroot to node 1 is supposed to be dropped in the trial solution. The other possible trial solution adds an edge $(3,2)$ and drops the edge $(1,2)$ thus leaving the initial tour uneffected. Observe, in the latter case, edge $(1,2)$ has been added at the very beginning and will be examined to be dropped once again at the end of the procedure.

From Fig. 1(b) we obtain a modified reference structure through a 1-exchange. Either we add an edge from tip 3 to a node of the stem (Fig. 1(c)), or we add an edge connecting tip 3 to a node of the cycle (Fig. 1(d)). In Fig. 1 edge $(3,4)$ indicates this step. In both cases, a unique edge $(4,5)$ must be dropped in order to receive a modified stem-and-cycle reference structure. Hence, node 5 becomes the new tip. Once again a trial solution can be obtained connecting node 5 to one of the subroots, either the non-labeled subroot or node 2 in Fig. 1(c). There are two trial solutions in Fig. 1(d) both of which do not affect node 2. Node 2 is no longer a subroot or member of the cycle. (In Fig. 1(d) adjacency of node 5 to one subroot is indicated only).

We emphasize again two features of the stem-and-cycle reference structure: (1) there are always two trial solutions available that create feasible tours, determined by the identity of the tip and the two subroots; (2) the subroots, which constitute two of the three nodes adjacent to the root, may change their identity after a move is executed, see Fig. 1(d).

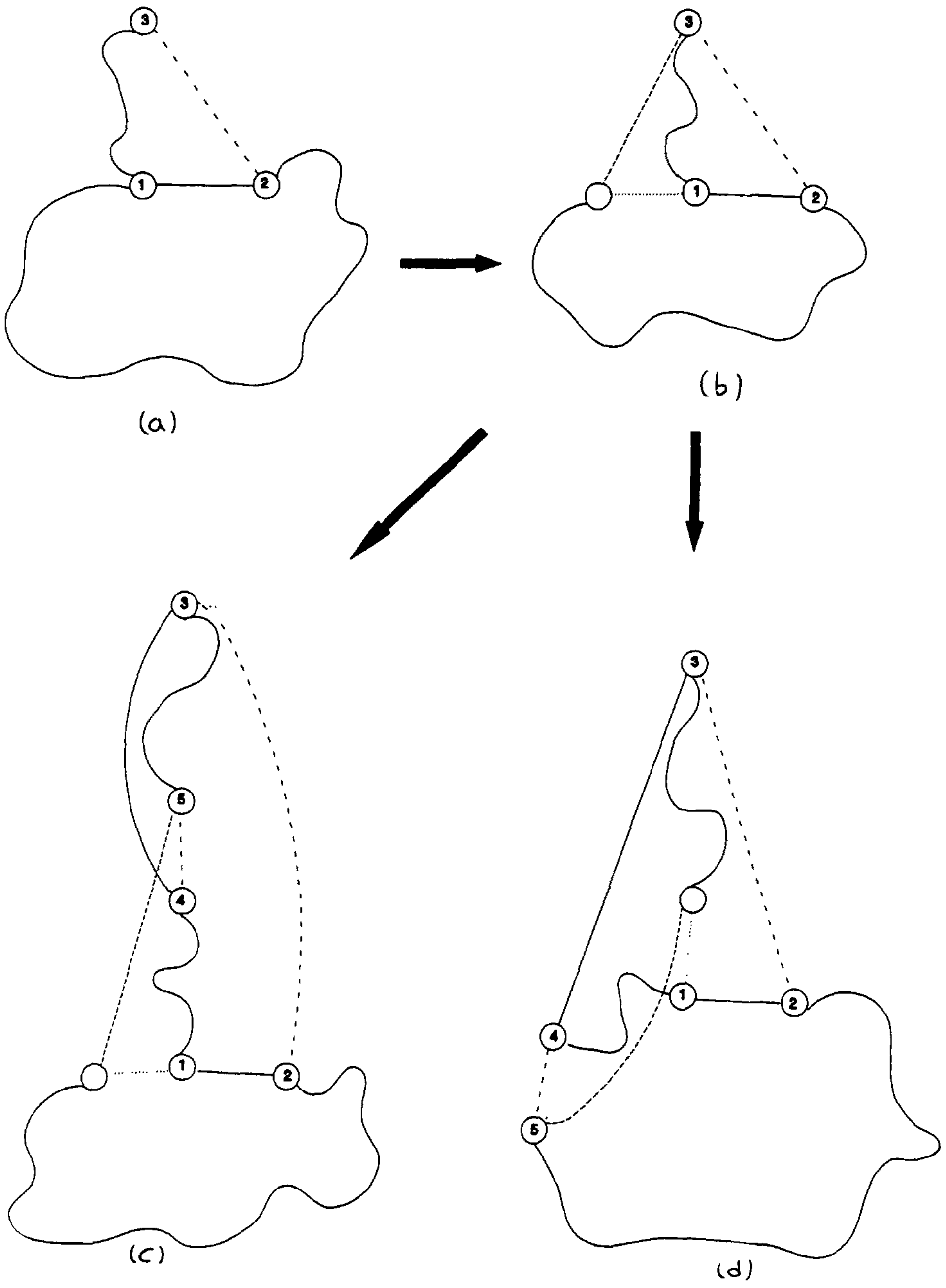

Fig. 1. An ejection chain based on 1-exchanges and a stem-and-cycle reference structure.

The rules we have indicated for admissible moves are not the only ones possible, but they are important for our present development because they maintain an ejection chain that corresponds to an alternating path. Later we will disclose special implications of these rules and the tours they can generate.

4. Variable depth format

To specify an ejection chain approach based on the stem-and-cycle reference structure, we first indicate a restricted version of the approach that follows the classical *variable depth* format outlined in [15]. The main feature of this format is to lock each part of the construction into place, so that no element (edge) added can ever to be removed, and no element removed can ever be added. This continues until the construction finally can proceed no further, and the process is restarted from scratch.

In referring to this format, we introduce terminology commonly used in tabu search to assign names to the lists that lock edges in and out of solution. Then we will later show how we can alter the functions of these lists to conform more closely to the functions used in tabu search, and endow them with a dynamic aspect that can create various *localized* versions of a tabu search procedure. By an extremely simple instance of this approach, we will show that we are able to create ejection chains for the TSP that have the properties identified in the theorems of Glover [6].

Our approach can readily be embedded in a complete tabu search implementation, or in a genetic algorithm or simulated annealing implementation (cf. [8, 10]). In the present study, however, we use such a method only as a stand-alone heuristic, which terminates when it is unable to find an improved solution at the conclusion of any of its constructive passes. (This follows the customary format of a variable depth procedure.)

As mentioned above a 2-exchange is a minimal move with respect to the TSP, i.e., there is no other feasible move affecting less than 2 edges (or, equally, a move that leads to a feasible solution). Actually the 2-exchange is a compound move of two 1-exchanges each of which by itself is not feasible. The first one introduces an edge in the current tour and ejects another one leading to an infeasible solution (only the number of edges is maintained). The second one removes the infeasibility without affecting the two edges involved in the first 1-exchange. Generalizing this idea, there is a sequence or ejection chain of 1-exchanges in which each is incomplete, each subsequence leads to an infeasible solution, and a final 1-exchange produces a new feasible solution. The whole chain of 1-exchanges may be considered as a special type of r-exchange such that a sequence of moves (e.g., 1-exchanges) is compressed into a single compound move. The component moves (1-exchanges) carried forward from one level to the next are designed so that they maintain a reference structure as previously indicated – i.e., a structure that allows one or more feasible solutions to be recovered by straightforward and well defined steps. The stem-and-cycle reference structure we employ here additionally maintains the construction "close to" a feasible solution. In particular, we require only a final 1-exchange to transform the infeasible tour of the reference structure into a feasible tour. The stem-and-cycle structure admits two such transformations for recovering infeasibility.

As our construction proceeds, we therefore note the *trial solutions* (feasible tours) that would result by applying these feasibility-recovering transformations after each step, keeping track of the best. At the conclusion of the construction we simply select this best trial solution to replace the current tour, provided it yields an improvement. In this process, the moves at each level cannot be obtained by a collection of independent and non-intersecting moves of previous levels. The list of forbidden (tabu) moves grows dynamically during a variable depth search iteration and is reset at the beginning of the next iteration.

We designate the lists of edges locked in and out of the solution by the names *tabu-to-drop* and *tabu-to-add*, where the former contains edges added by the current construction (hence which must be prevented from being dropped, except in the case of creating a trial solution) and the latter contains edges dropped by the current construction (hence which must be prevented from being added except in the case of a trial solution).

The resulting ejection chain procedure is shown in Fig. 2. We denote the cost of a tour T by $c(T)$. The reference structure that results by performing d paired ejection steps (i.e., 1-exchanges in the present case), is denoted by $T(d)$, where d is the "depth" of the ejection chain (hence $T = T(0)$ for a given starting tour T).

The above procedure describes in its inner *repeat* … *until* loop one iteration of an ejection chain search. The *while* … *do* loop describes one component move. Starting with an initially best solution $T^*(0)$, the procedure executes a construction that maintains the reference structure for a certain number of component moves. The new

```
begin
    Start with an initial solution T*.
    T := T*;
    Let s be a city in T.        {s is the root}
    k* := s;
    repeat
        d := 0;  {d is the current search depth}
        initialize the lists tabu–to–drop and tabu–to–add, e.g. empty lists;
        while   there are edges in T(d) that are not tabu–to–drop and edges outside of
                T(d) that are not tabu–to–add do
            i := k*;
            d:= d + 1;
            Find the best component move that maintains the reference structure, where this 'best'
            is given by the edge pair (i,j*), (j*,k*) for which the gain
            $g(i,j^*,k^*) = \max \{g(i,j,k) \mid g(i,j,k) = c_{jk} - c_{ij}$ ; (i,j) is not an edge in T(d–1) and
            (j,k) is an edge in T(d–1); (i,j) is not tabu–to–add; (j,k) is not tabu–to–drop}; {Note
            that g(i,j*,k*) can be negative.}
            Perform this move, i.e. introduce edge (i,j*) and remove edge (j*,k*) thus obtaining
            T(d) as a new reference structure at search depth d;
            (i,j*) becomes tabu–to–drop and (j*,k*) becomes tabu–to–add;
            Let s' be that neighbor of s in T(d) such that the component move which ejects the
            non–tabu–to–drop edge (s',s) and inserts the edge (k*,s') yields a largest gain; let
            T*(d) denote the preferred associated trial solution.
        Let d* denote the search depth at which the best solution T*(d*) with
        $c(T^*(d^*)) = \min \{c(T^*(d) \mid 0 < d \le n\}$ has been found;
        if d* > 0 then begin T* := T*(d*); T := T* end
    until d* = 0;
end
```

Fig. 2. An ejection chain procedure based on a variable depth format with respect to 1-exchange component moves for maintaining the reference structure.

currently best trial solution $T^*(d^*)$, encountered at depth d^*, becomes the starting point for the next ejection chain iteration. The iterations are repeated as long as an improvement is possible. The maximum depth of the construction is reached if all edges in the current solution T are set tabu-to-drop. The step leading from a solution T to a new solution consists of a varying number d^* of component moves, hence motivating the "variable depth" terminology. A continuously growing tabu list avoids cycling of the search procedure. As an extension of the algorithm, the whole *repeat* ... *until* part could easily be embedded in yet another control loop (not shown here) leading to a multi-level (parallel) search algorithm, see [6].

The algorithm of Lin and Kernighan [13] (or Mak and Morton [14]) is a special instance of the above procedure, where the neighbor city s' of the starting city s is always the same. By contrast, in the procedure of Fig. 2, both neighbors of s in T may be considered to take the role of the "last visited" city s'. (That is, the stem-and-cycle reference structure provides a more flexible set of alternatives than the reference structure implicit in these earlier methods. Additional consequences of this will be noted shortly.) Fig. 1 illustrates the procedure for two 1-exchanges. The labels of the cities describe the execution. Starting with city 1 there is an edge added to city 2 which ejects the existing edge (2,3). After a feasibility and improvement check from 3 to a neighbor of 1 a new edge connecting 3 to 4 is introduced (both cases are indicated, either 4 lies on the path part of the stem-and-cycle reference structure, or 4 lies on the cycle part). Edge (4,5) is ejected and a final 1-exchange connects 5 to one of the two neighbors of s in x.

Note, the ejection chain procedure of Fig. 2 can also be used as a tour construction procedure in terms of a sequence of node insertions. In that case, the first step is slightly modified, where in the initial tour edge (i, j^*) is inserted and edge (j^*, k^*) is deleted. Instead of connecting i to j^*, both nodes become neighbours of the new node v to be inserted. The remaining steps of the ejection chain procedure are the same. Thus, Fig. 2 used as a construction algorithm, also covers the node insertions (types I and II) and node deletions (types I and II) described in [4]. It is sufficient to consider node i as the root and node $i+1$ as the tip of the initial stem-and-cycle reference structure. After the first three steps of the ejection chain procedure of Fig. 2 the same edge exchanges can be obtained as presented in [4].

5. A localized tabu search format

As noted, the preceding variable depth format provides an opportunity to make moves that are inaccessible to the Lin–Kernighan procedure, due to our incorporation of the stem-and-cycle reference structure within it. However, we can do better than this by allowing a more general format that does not so narrowly constrain the use of the "tabu" lists, which currently lock moves rigidly into place. For motivation, we note that tabu search typically uses tabu lists whose composition changes in an adaptive manner. Elements in general are assigned a *tabu status* for creating penalties or probabilities that govern their selection. Such a status depends on aspects such as the recency and frequency in which given elements are added to the list.

The management of these memory structures in tabu search also integrates short term and long term designs, permiting the search to continue beyond points of local optimality where a current construction fails to produce an improved solution. Variable depth procedures are also sometimes claimed to have an ability to go beyond local optimality. This occurs in the restricted sense that such a method may generate trial solutions during its constructive pass that are inferior to the best recorded trial solution, and an inferior trial solution is loosely construed as a "local optimum" if it is better than one examined immediately before or after it. (Of course, any method that scans a set of alternative solutions on a given pass and retains only the best has this same feature).

In the present case we will embed a simple form of TS memory only within the framework of an isolated pass of the variable depth format. This approach will not include uses of memory to coordinate successive passes, and consequently we will refer to it as a *localized* TS application. This limited form of tabu search nevertheless yields interesting consequences.

5.1. Dynamic alternating paths

Within the format of the variable depth approach, which prevents an edge from being reused, the stem-and-cycle is able to generate a construction that satisfies the usual graph theory definition of an alternating path. In particular, each edge removed belongs to T and each edge added belongs to $E-T$. However, it is useful to generalize this definition to consider a *dynamic* variant in which the path itself can contribute edges to add and to remove. Such a generalization is a step in the same direction as that of introducing the type of memory used in tabu search.

By restricting attention to a localized TS approach, we can significantly limit the type of dynamic alternating path considered. This possibility arises due to a theorem about dynamic path generated by the stem-and-cycle structure. In particular, starting from an arbitrary tour T, it is possible to visit any other specified tour by a special dynamic alternating path called a *delete–add simple* path. Such a path is defined by stipulating that no edge deleted is added back, while an edge that is added may be deleted, but at most once – hence the term "simple".

The lists used by the variable depth format can be modified to permit a localized TS process for generating just these delete–add simple paths. The modification is remarkably simple: it suffices simply to discard the *tabu-to-drop* list. The reason is as follows. Preventing deleted edges from being added back implies that the *tabu-to-add* list is retained, but once an added edge is allowed to become deleted, it will in fact become a member of the tabu-to-add list, and thus it can never be deleted (or added) again. From a strategic standpoint, such a simple restricted memory may not be best, but it invites examination due to its compatability with the theorem that specifies the existence of a path from T to any other tour.

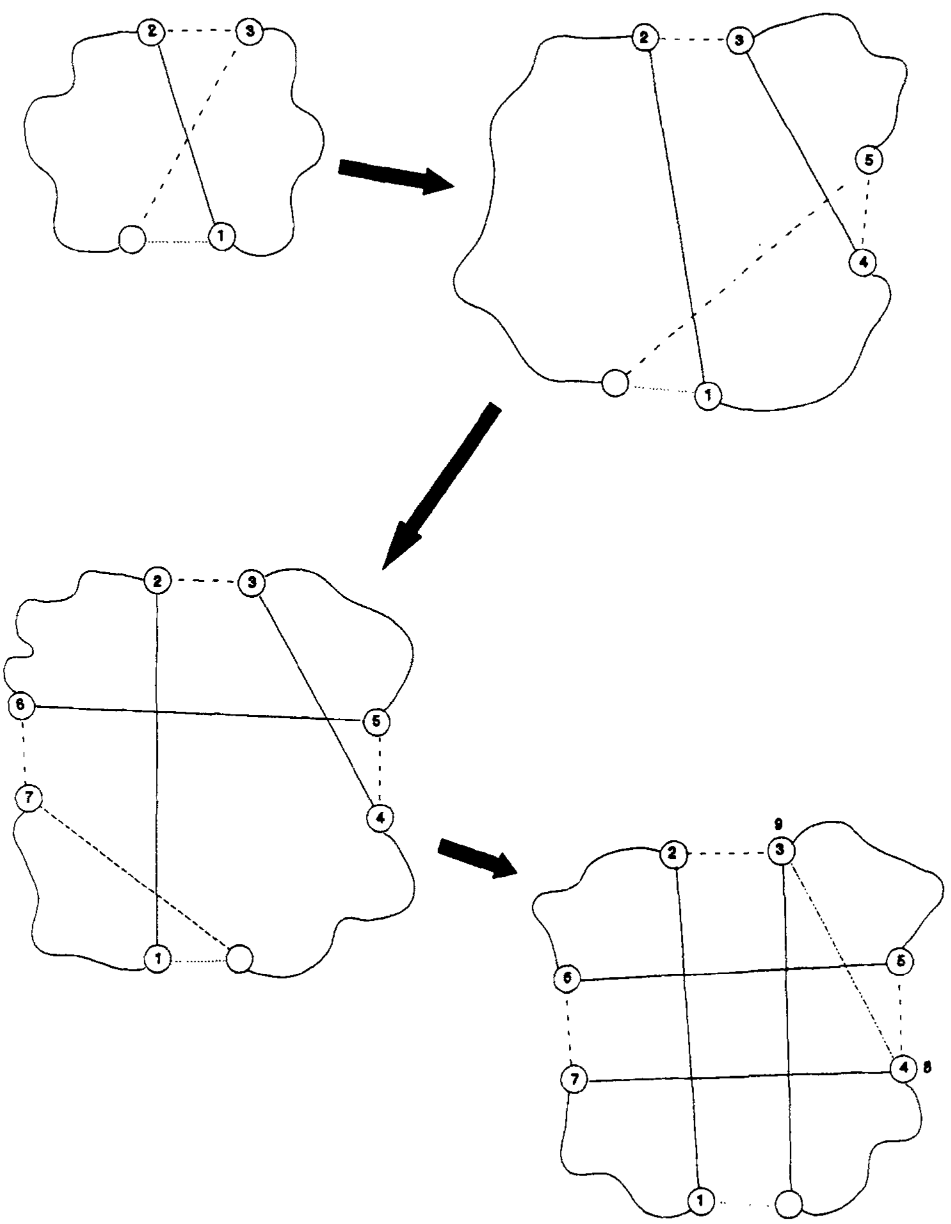

Fig. 3. A limited 4-opt case.

An example of an interesting construction made possible by the stem-and-cycle structure using this memory is shown in Fig. 3. In this case by allowing a single added edge to be deleted on a later step, a special type of 4-exchange is produced by the procedure of Fig. 2. This exchange is unattainable by an application of the Lin–Kernighan procedure, and is sometimes introduced to supplement this procedure in an effort to improve its performance. This is illustrated in Fig. 3 where an edge $(3,4)$ inserted in an earlier step is removed again (see labeling 8,9 in Fig. 3). Finally, the compound move leading to an infeasibility is to replace edges $(2,3)$, $(4,5)$, and $(6,7)$ by the new edges $(1,2)$, $(5,6)$, and $(7,4/8)$. A final component move guarantees a feasible outcome. Thus the procedure of Fig. 2 can easily be implemented to include also the limited 4-opt neighborhood which is not accessible to the basic Lin–Kernighan approach. In addition, the procedure automatically incorporates many other neighborhoods beyond the scope of the Lin–Kernighan approach.

6. Other TSP ejection chain methods

We have so far focused on a single type of ejection chain approach for the TSP. The principles underlying this approach also give a foundation for implementing other types of ejection chain methods. Thus, for example, we can make use of *node ejection* and *subpath ejection* moves [5, 6]. In these constructions an initial node (or subpath) is moved to occupy a new location in the tour, thus ejecting a node (or subpath) from that location, and so forth, in a string of ejections that ultimately ejects the element that initiated the first ejection. Alternately, the process can begin by a *capping move* and terminate by an *anchoring move*, so that the initial element is not ejected by the last, but by an edge which joins the nodes that lie on either side of the first ejected element, while the last element is simply inserted between two nodes that are currently adjacent. Fig. 4 illustrates an ejection chain based on node ejection moves, in the case where the last node ejects the first.

A component move consists of replacing a node, j say, by a node i such that the neighbors j' and j'' of j become new neighbors of i. The reference structure consists of a path connecting the former neighbors i' and i'' of i via the edges (j',i) and (i,j''), and the isolated node j. A check on improvement and feasibility makes the node j adjacent to i' and i''. The chain continues node j ejects a node, say k, from its current tour position, i.e., (k',j) and (j,k'') become new edges within the modified reference structure and node k is the newly isolated node. The procedure continues until in the last step of an ejection chain iteration the reference structure is transformed into a feasible tour. That means the finally isolated node becomes a new neighbor of nodes i' and i''. Fig. 4 illustrates an ejection chain based on node exchange moves. Eight new edges are included into the new solution, namely $(1',2),(2,1''),(2',3),(3,2''),(3',4),(4,3''),(4',1)$, and $(1,4'')$. Eight edges of the old tour are deleted, namely $(1',1),(1,1''),(2',2),(2,2''),(3',3),(3,3''),(4',4)$, and $(4,4'')$.

There are a number of possible modifications of ejection chains even with respect to one particular neighborhood structure. The interested reader is referred to [5, 6]. One may also think of modifications with respect to the component moves. Each component move might be considered as one step in a simple local search procedure which is not necessarily based on feasible solutions but on reference structures closely related to feasible solutions. A component move is performed greedily avoiding tabu moves.

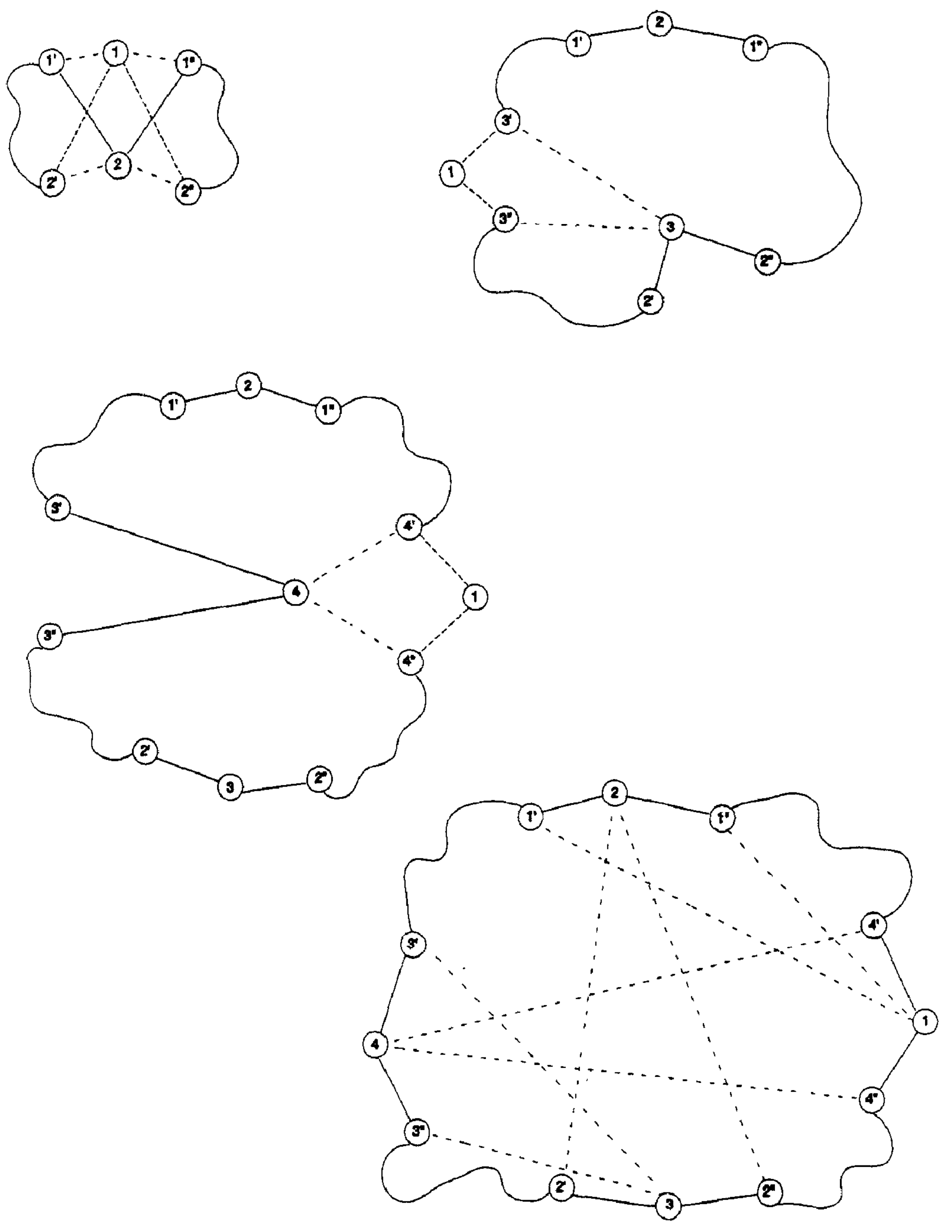

Fig. 4. An ejection chain based on node exchange moves.

A probabilistic acceptance (as in simulated annealing or probabilistic tabu search) is also possible. Likewise, ejection chains can easily be included in a genetic algorithm framework by using them as improvement procedures to supplement the genetic approach.

7. Computational results

The particular variable depth ejection chain procedures we have chosen for examination are implemented in PASCAL and have been tested on a VAX 8650 under VMS. The same coding and implementation environment has been used in the study of genetic algorithm and simulated annealing approaches by Ulder et al. [20], providing a compatible basis for comparing results. Likewise, we use the same test bed instances as in [20], ranging from 48 to 666 cities, as indicated in Table 1. We also adopted the running time limit used in [20] for these particular instances. Table 1 presents the results over number of search processes, i.e., restarts (indicated in brackets) that begin from randomly chosen initial feasible tours. The results are expressed by identifying the average tour lengths of the final (best) solutions generated from the successive runs. The percentages shown in the table are the percentage deviations of these average "best tour lengths" above the length of an optimal solution.

The columns Mult2-Opt and MultLK give the results of Ulder et al. [20] which have been obtained for a 2-opt local search and the Lin–Kernighan procedure. Column MultLK2 contains results of a modified Lin–Kernighan procedure. It is based on the observation that optimal or near-optimal solutions often can be constructed for traveling salesman problems by limiting consideration to a small number of shortest edges incident to each node (for instance 5 to 20 depending on the problem size). Hence moves are performed on a neighborhood that is restricted in a certain sense, i.e., the set of candidate moves is reduced to the preferred attribute candidate list. Only those move candidates are considered that meet a preferred attribute, e.g., including some of the shortest edges. Such moves can be classified by their amplification factor which is the number of edges added by a move devided by the number of edges that belong to a "k-shortest" category. The Lin–Kernighan modification as described in column MultLK2 of Table 1 is as follows.

(1) Randomly divide the node set in subsets (not necessarily non-overlapping) of 50 to 100 nodes (respectively cities). Consider $n/10$ to $n/5$ subsets where n is

Table 1
Computational results

Instance	t	Mult2-Opt	MultLK	MultLK2	MultEC
GRO48	6	1.35 (40)	0 (19)	0 (19)	0 (16)
TOM57	10	1.34 (41)	0 (14)	0 (17)	0 (14)
EUR100	60	3.23 (67)	0 (31)	0 (33)	0 (24)
GRO120	86	4.57 (64)	0.08 (30)	0.02 (36)	0.05 (27)
LIN318	1600	6.35 (99)	0.37 (37)	0.21 (43)	0.11 (29)
GRO442	4100	9.29 (108)	0.27 (67)	0.09 (84)	0.04 (48)
GRO532	8600	8.34 (116)	0.37 (77)	0.28 (85)	0.10 (44)
GRO666	17000	8.67 (122)	1.18 (45)	0.82 (81)	0.41 (21)

number of the cities in the underlying problem instance. (Size and number of the subsets are chosen randomly.)

(2) Consider each subset of cities as a traveling salesman problem which is solved using the algorithm of Lin and Kernighan [13].

(3) The edges of all tours obtained in (2) define a preferred attribute candidate list. Solve the complete traveling salesman problem on these preferred edges using the algorithm from Lin/Kernighan, i.e., in each step of a Lin–Kernighan iteration an edge is allowed to be introduced into the currently modified tour only if it belongs to the set of preferred edges. In the final step of a Lin–Kernighan iteration (the step that achieves a feasible tour) an arbitrary edge may be chosen to be introduced into the newly generated tour.

Table 1 shows that MultLK2 did not create a subproblem for the smallest problem instance consisting of 48 cities. In all remaining cases the number of runs of MultLK2 starting from a randomly generated initial solution is, particularly for the larger instances, substantially higher than for MultLK. This is an effect of the restricted number of tests for the component moves that have to be performed during the search process, because the number of edges is enormously reduced. The quality of the outcome is also better. A possible reason for the improved outcome is not only an effect of inheriting edges from the suboptimal solutions of the subproblems. We conjecture that a main reason for the improved outcome is the diversification effect of the preferred edge candidate list. The restricted number of edges available for the local (infeasible) moves within a Lin–Kernighan iteration may force the method temporarily to take steps that are less attractive than those available in the presence of more edges, including steps that cause greater deteriorations, while the overall achievement of the Lin–Kernighan iteration compensates for these deteriorations. Alternatively, the improved outcomes may result simply from the effect of restricting consideration to a subspace in which the combinatorial alternatives are greatly diminished in number.

The last column MultEC indicates the results obtained with an edge based ejection chain procedure as described in Section 4. Within a complete run only one edge was allowed to be removed from the tabu list tabu-to-drop after it had been inserted earlier. We can see that the number of runs of the ejection chain approach, within the time limit adopted drops almost to half the number of MultLK or MultLK2 runs. This presumably is an effect of the expanded search space resulting from the increased number of possible local exchange steps. The average tour lengths (over the number of restarts within the adopted time limits) of MultEC are the best obtained from all considered methods. Even for the biggest instance with 666 cities the worst (average) tour length is less than half percent from the length of a shortest tour.

The two ejection chain methods we implemented are significantly the best of all methods tested, yielding notably smaller deviations from the optimal solutions found by all methods. In addition to the ejection chain methods whose results are recorded in these tables, we also tested other types of ejection chain methods which did not prove nearly as effective. These included methods based on node ejections (as described earlier), subpath ejections (considering subpaths of at most 3 edges), and edge ejections based on other reference structures that were "further from" feasibility (requiring more steps to recover a feasible tour). For instance, our implementation of node based ejection chains, even using a preferred attribute candidate list, could not obtain tour lengths better than 10% above the optimum tour length. A possible reason for this behavior is that the ejection chains at the focus of this study consist of minimal steps; that is, local 1-exchanges represent minimal modifications of the reference structure. Node ejections induce a more appreciable change in the underlying reference structure, and hence are not so well suited for fine tuning as edge ejections. On the other hand, Rego and Roucairol [17] report good success with node ejections for vehicle routing problems. To what extent this is attributable to inherent differences between such problems and TSPs, or to differences in implementation, is unknown at present. It is clear, however, that the edge based ejection chains, that use the stem-and-cycle reference structure are highly effective for the TSP.

8. Conclusions section

We have presented an ejection chain algorithm for the traveling salesman problem that performs substantially better than other local search procedures considered to represent the current state of the art – proving superior to recent variants of the Lin–Kernighan procedure and extensions involving genetic algorithms and simulated annealing. Ejection chains afford a significant potential for incorporating different types of local search learning structures. They can be used either as local improvement procedures to accelerate "evolutionary approaches" such as genetic algorithms, or as general search strategies for guiding local exchange steps at each iteration. Each iteration similarly can be implemented within the framework of simulated annealing or tabu search.

A noteworthy feature of ejection chains lies in their general applicability. Ejection chains can be based in a natural way on edges, nodes, subpaths, alternations of edges and nodes, and so forth, depending on the reference structure and type of moves adopted. Reference structures that embrace infeasible solutions provide a much greater flexibility for defining effective moves. They are particularly useful for developing strong compound moves – moves that may implicitly operate within a search space greatly larger than their immediate neighborhood search space. The ultimate potential of ejection chains is largely unknown, and invites further examination from both theoretical and applied perspectives.

Acknowledgements

The authors are grateful to the two anonymous referees for their constructive comments.

References

[1] D. Applegate, R. Bixby, V. Chvatál and W. Cook, Finding cuts in the TSP, Preliminary Report, At&T Bell Labs, Rice University, Rutgers University and Bellcore (1994).
[2] C. Berge, Theory of Graphs and its Applications (Methuen, London, 1962).
[3] U. Dorndorf and E. Pesch, Fast clustering algorithms, ORSA J. Comput. 6 (1994) 141–153.
[4] M. Gendreau, A. Hertz and G. Laporte, New insertion and postoptimization procedures for the traveling salesman problem, Oper. Res. 40 (1992) 1086–1094.
[5] F. Glover, Multilevel tabu search and embedded search neighborhoods for the traveling salesman problem, Graduate School of Business, University of Colorado, Boulder (1991).
[6] F. Glover, Ejection chains, reference structures and alternating path methods for the traveling salesman problem, Graduate School of Business, University of Colorado, Boulder (1992).
[7] F. Glover and E. Pesch, Ejection chains and graph-related applications, Graduate School of Business, University of Colorado, Boulder (1995).
[8] D.S. Johnson, Local optimization and the traveling salesman problem, Proceedings of the 17th Colloquim on Automata, Languages, and Programming (Springer, Berlin, 1990) 446–461.
[9] J.P. Kelly, M. Laguna and F. Glover, A study of diversification strategies for the quadratic assignment problem, Comput. Oper. Res. 21 (1994) 885–893.
[10] A. Kolen and E. Pesch, Genetic local search in combinatorial optimization, Discrete Appl. Math. 48 (1994) 273–284.
[11] M. Laguna, J. Kelly, J.L. Gonzales-Velarde and F. Glover, Tabu search for the multilevel generalized assignment problem, Graduate School of Business, University of Colorado, Boulder (1991) European J. Oper. Res., to appear.
[12] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, The traveling salesman problem (Wiley, Chichester, 1985).
[13] S. Lin and B.W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Oper. Res. 21 (1973) 498–516.
[14] K.-T. Mak and A.J. Morton, A modified Lin–Kernighan traveling-salesman heuristic, Oper. Res. Lett. 13 (1993) 127–132.
[15] C.H. Papadimitriou and K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity (Prentice-Hall, Englewood Cliffs, NJ, 1982).
[16] E. Pesch, Learning in Automated Manufacturing (Physica, Heidelberg, 1994).
[17] C. Rego and C. Roucairol, An efficient implementation of ejection chain procedures for the vehicle routing problem, Research Report RR-94/44, PRISM Laboratory, University of Versailles (1994).
[18] G. Reinelt, The Traveling Salesman (Springer, Berlin, 1994).
[19] W.R. Stewart, J.P. Kelly and M. Laguna, Solving vehicle routing problems using generalized assignments and tabu search, Graduate School of Business, University of Colorado, Boulder (1994).
[20] N. Ulder, E.H.L. Aarts, H.-J. Bandelt, P.J.M. van Laarhoven and E. Pesch, Genetic local search algorithms for the traveling salesman problem, Lecture Notes in Computer Science, Vol. 496 (1991) 109–116.
[21] J. Xu and J.P. Kelly, A robust network flow-based tabu search approach for the vehicle routing problem, Graduate School of Business, University of Colorado, Boulder (1995).