ELSEVIER Discrete Applied Mathematics 65 (1996) 223–253

DISCRETE APPLIED MATHEMATICS

Ejection chains, reference structures and alternating path methods for traveling salesman problems[☆]

Fred Glover*

School of Business, Campus Box 419, University of Colorado, Boulder, CO 80309, USA

Received 4 January 1993; revised 12 July 1994

Abstract

Ejection chain procedures are based on the notion of generating compound sequences of moves, leading from one solution to another, by linked steps in which changes in selected elements cause other elements to be "ejected from" their current state, position or value assignment.

This paper introduces new ejection chain strategies designed to generate neighborhoods of compound moves with attractive properties for traveling salesman problems. These procedures derive from the principle of creating a reference structure that coordinates the options for the moves generated. We focus on ejection chain processes related to alternating paths, and introduce three reference structures, of progressively greater scope, that produce both classical and non-standard alternating path trajectories. Theorems and examples show that various rules for exploiting these structures can generate moves not available to customary neighborhood search approaches. We also provide a reference structure that makes it possible to generate a collection of alternating paths that precisely expresses the symmetric difference between two tours.

In addition to providing new results related to generalized alternating paths, in directed and undirected graphs, we lay a foundation for achieving a combinatorial leverage effect, where an investment of polynomial effort yields solutions dominating exponential numbers of alternatives. These consequences are explored in a sequel.

Keywords: Traveling salesman; Graph theory; Combinatorial optimization; Integer programming; Neighborhood search

1. Introduction

Ejection chain strategies give a useful way to build compound neighborhoods, with the goal of generating more powerful moves for solving discrete optimization

☆ This research was supported in part under the Air Force Office of Scientific Research and Office of Naval Contract F4-9620-90-C-0033 at the University of Colorado.

* E-mail: fred.glover@colorado.edu.

SSDI 0166-218X(94)00037-2

problems. Ejection chains combine and generalize ideas from a number of sources, including classical alternating paths from graph theory [1, 14], network related base exchange constructions in matroid optimization [5, 17], and bounding form structures for solving integer programming problems [7]. Each of these embodies a related theme whose incorporation into neighborhood search offers a variety of new approaches to combinatorial optimization applications.

In rough overview, an ejection chain is initiated by selecting a set of elements to undergo a change of state (e.g., to occupy new positions or receive new values). The result of this change leads to identifying a collection of other sets, with the property that the elements of at least one must be "ejected from" their current states. 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). In some cases, a cascading sequence of operations may be triggered representing a domino effect. The ejection chain terminology is intended to be suggestive rather than restrictive, providing a unifying thread that links a collection of useful procedures for exploiting structure, without establishing a narrow membership that excludes other forms of classification.

A number of methods deriving from this perspective recently have appeared in the literature. A *node (or block) ejection* procedure has been proposed by Glover [8] for traveling salesman problems, and extended to provide new approaches for quadratic assignment and vehicle routing problems. Laguna et al. [16] introduce an ejection chain approach in conjunction with a tabu search procedure for multilevel generalized assignment problems, and demonstrate that ejection chains even of "small depth" produce highly effective results in this context. Ejection chain strategies are also proposed for clique partitioning by Dorndorf and Pesch [3] and for other forms of clustering problems by Hubscher and Glover [15], similarly yielding good outcomes.

This paper focuses on the traveling salesman problem (TSP), whose goal is to find a minimum cost Hamiltonian cycle (tour) on a graph of n nodes. Letting $c(i,j)$ denote the cost of edge (i,j), and allowing the notational convention of treating paths and cycles as edge sets, the objective may be expressed as seeking a tour T that minimizes $\sum(c(i,j) : (i,j) \in T)$. (See [18] for a comprehensive background.)

We characterize ejection chain strategies for the TSP that are founded on the notion of creating a reference structure to guide the generation of acceptable moves. We show such a structure can be controlled to produce transitions between tours with desirable properties, in particular generating alternating paths (or collections of such paths) of a non-standard yet advantageous type.

The organization of the paper is as follows. We begin by identifying a basic stem-and-cycle reference structure in Section 2, and then describe a Subpath Ejection Method for exploiting it in Section 3. We also illustrate how the approach generates tours that cannot be obtained by "connectivity preserving" methods such as the Lin–Kernighan procedure. Fundamental rules for treating stem-and-cycle reference structures are introduced in Section 4, together with characterizations of various types of alternating paths that differ from those customarily treated in graph theory literature. Section 5 examines parallel processing options applicable to these methods.

In Section 6 we introduce a more general doubly rooted reference structure, with examples and theorems to characterize advantages provided in exchange for a relatively modest increase in computational overhead. Section 7 completes the hierarchy of reference structures with a stem-and-multicycle structure capable of generating precisely the symmetric difference between any two tours, producing a succession of alternating path trajectories that never add or drop a "wrong" edge. Finally, Section 8 examines implications of these outcomes, and identifies recent theoretical extensions and empirical studies that indicate the computational value of these results.

2. A stem-and-cycle reference structure

The stem-and-cycle reference structure is a spanning subgraph that consists of a node simple cycle attached to a path, called a stem. The node that represents the intersection of the stem and the cycle is called the root node, denoted by r, 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 t. An example of the stem-and-cycle structure is shown in Fig. 1.

The stem can be degenerate, consisting of a single node, in which case $r = t$ 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

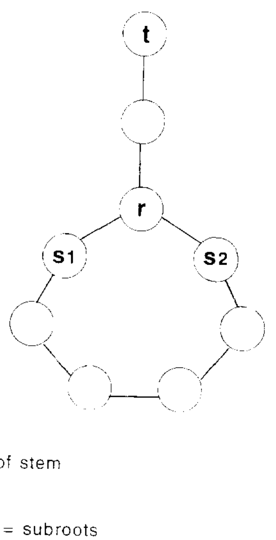


Fig. 1. Stem-and-cycle reference structure.

(t, s) from the tip to one of the subroots s, and deleting the edge (r, s) between this subroot and the root. (When the stem is degenerate, this operation adds and deletes the same edge, leaving the tour unaffected.)

3. Chains for ejecting tour subpaths

The first ejection chain construction we consider operates by cutting out and relinking tour subpaths. The process generates an evolving stem-and-cycle configuration which is initiated by selecting a node of the current tour to be the root node r, and hence also the initial tip t. The stem is then extended by a series of steps that consist of attaching a new non-tour edge (t, j) to a chain of tour nodes $(j, \ldots, k)$ which is thereby ejected from the tour. We forbid the ejected segment from including any nodes from a critical collection which initially contains node r. In the simplest case, this segment can consist of only node j itself (i.e., $j = k$). Pesch [21] has suggested a variant of the node ejection of Glover [8] that follows a similar design, although without reference to subpaths or the guiding influence of the stem-and-cycle structure. The general form of the method is as follows.

Subpath Ejection Method.

Step 1: Select the root node r and let $t = r$. Let PC and SC respectively denote the sets of primary and secondary critical nodes, where to begin PC $= \{r\}$ and SC is empty. Designate both subroots to be *accessible*.

Step 2: Identify a tour subpath $(j, \ldots, k)$ that contains no nodes of PC or SC. Eject this segment from the tour by adding edge (t, j) and dropping the two tour edges (j', j) and (k, k'') that lie outside the subpath (where, relative to the orientation of the subpath, j' may be viewed as the predecessor of j, and k'' may be viewed as the successor of k). Complete the step by adding the non-tour edge (j', k'') to "cap the hole" left by the extracted segment.

Step 3: Let $t = k$, identifying the new tip of the extended stem. Add nodes j and k to PC, and nodes j' and k'' to SC. If a node added to PC is a subroot, then designate it to be no longer accessible and add the remaining subroot also to PC (to preserve its accessibility).

Step 4: Examine the trial solutions for creating a completed tour that result by joining t to each accessible subroot (keeping a record to recover the best trial solution found). Stop if PC and SC together contain all tour nodes (or if the quality of the trial solutions has not attained a desired threshold for a chosen number of steps). Otherwise return to Step 2.

We note the preceding method permits subpaths to be ejected within other subpaths previously ejected. The sets PC and SC governing legitimate ejections need not be disjoint (a node can belong to both if it is added to PC first), but only their union is relevant except in the special case where a subroot becomes a member of PC. The method also allows one "exceptional situation." We have not required that the edge joining t to a subroot, for creating a trial solution, must be a non-tour edge. It is possible that this edge may be a tour edge that was deleted immediately prior to examining the trial solution. Although the edge has been temporarily deleted and then added back, the organization of the method encounters no contradiction by this.

Throughout this paper we assume that data structures for carrying out prescribed operations are evident. A simple example will be given for the present method. Begin the SEM procedure with a list J that contains the nodes in any order. Associated with J keep an array LOC(i) that names the position of node i on J (i.e., $J(\mathrm{LOC}(i)) = i$, where $\mathrm{LOC}(i) = 0$ if i is not on J, or LOC(i) can simply be a binary flag in this limited application). When a node thus is to be added to PC or SC, it is removed from J, as by writing the last node of J over it and decreasing the size of J by 1 (updating the LOC array appropriately). Then at each execution of Step 2, node j can be any element of J, and the subpath (j, k) can be generated by tracing along the tour in either direction from node j, stopping when desired, subject to not going beyond the first node i encountered such that $\mathrm{LOC}(i) = 0$. In the latter case, the node reached just before i is the last acceptable choice of k in the direction traced.

By this organization, any choice rule that allows at most a fixed given number of nodes to be examined at each iteration of Step 2 (in selecting j from J and tracing the subpaths (j, k) as indicated) will result in an effort of $O(n)$ for applying the method from start to finish.

We now establish the validity of the method as follows.

Proposition. *Each trial solution generated at Step* 4 *of the Subpath Ejection Method is a feasible tour.*

Proof. We must show that the method generates a valid stem-and-cycle structure at each step, and that a subroot qualifies as accessible only if in fact it constitutes a current subroot. By induction, assume the construction gives a valid stem-and-cycle structure before the current execution of Step 2. If no node of $(j, \ldots, k)$ is on PC or SC, then none of these nodes meets a previously added or deleted edge, and hence the subpath is well defined, constituting a component of the initial tour. Moreover, (j', j) and (k, k'') also must be edges of the initial tour. If the ejected segment comes from the current cycle, the capping edge therefore creates a new cycle, while if it comes from the current stem, the capping edge retains the stem intact (where in each case the ejected segment is transferred to the tip). Finally, since the root r belongs to PC, no new non-tour edge can meet r (after the first) except where r also becomes a member of SC. This implies that an edge is dropped between the root and a subroot s and that s is added to PC (as j or k) in Step 4. Thus this condition accurately identifies that s no longer fulfills the role of a subroot, and the operation of adding the remaining subroot to PC assures it cannot become inaccessible. □

3.1. Exploiting the subpath ejection neighborhoods

The Subpath Ejection Method generates a compound neighborhood for defining moves to transition from a current tour to a new one. The termination condition in Step 4 provides a signal to select the best trial solution found at this step as the new current tour, and then to begin again with this tour at Step 1.

Our motive for creating an ejection chain of the type generated by the Subpath Ejection Method derives from the following consideration. If the current tour is not optimal, there must exist some node r and a sequence of nodes starting from r and leading finally back to an adjacent node, that will bring the current tour in closer correspondence to an optimal tour. Moreover, if the current tour is already "locally optimal" (by applying an approach such as 2-opt, for example), then it is likely that a number of subpaths of the tour are already properly sequenced. Consequently, an ejection chain process designed to piece together appropriate segments may provide a foundation for finding additional improvements.

3.2. An illustrated construction

We provide an example to illustrate the kinds of outcomes the Subpath Ejection Method is capable of generating in transitioning from one tour to another, establishing a basis that leads to more advanced considerations. The graph for our example consists of ten nodes, shown in Fig. 2. The current tour is assumed to visit these nodes in their numerical order. The illustrated process ejects two subpaths, each consisting of a single edge, and then selects a trial solution. In Fig. 2, the ejection chain starts with node 1 as the root r, then adds edge $(1,8)$ to attach to and eject ("cut free") the subpath $(8,9)$ (which in this case consists of an edge), followed by adding edge $(9,4)$ to attach to and eject the subpath $(4,5)$ (likewise in this case consisting of an edge). Thus the capping edges added by these two ejections are $(7,10)$ and $(3,6)$, while the edges deleted are $(7,8)$, $(9,10)$, $(3,4)$ and $(5,6)$. Of the two trial solutions available at this point, we select the one associated with subroot 2 to conclude the process, thus adding edge $(5,2)$ and deleting edge $(1,2)$.

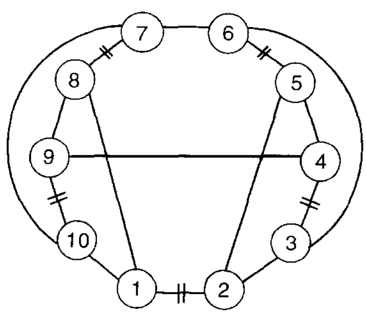


Fig. 2. An ejected subpath tour construction (not obtainable by maintaining tour connectivity).

The transformation of the current tour into the new one of this simple example produces an outcome that cannot be achieved by the popular heuristic approach of Lin and Kernighan [19], or by any heuristic that is designed similarly to maintain an underlying feasible tour construction at each step. Specifically, we see that the two deleted edges adjacent to the first added edge destroy the connectivity of the graph. That is, the initial addition of edge (1, 8) cannot be accompanied by the deletion of both (1, 2) and (7, 8), since this divides the graph into a disjoint path and subtour. Moreover, the same sort of infeasibility occurs in this example regardless of which edge is added first. Thus, the resulting tour cannot be obtained by a "connectivity preserving" approach regardless of which node of the graph is selected to initiate the process. (The Lin–Kernighan procedure allows for an optional maneuver on its first step in an attempt to overcome its connectivity limitation, but even after such a maneuver the example again destroys the connectivity required to continue at the following step.)

Additional somewhat different but equally simple constructions by the Subpath Ejection Method yield the same type of outcome. (For example, an instance of such a construction occurs by replacing the three interior edges of Fig. 2 by the three edges (1, 4), (5, 8) and (9, 2).) If it is indeed desirable to be able to build new tours by linking "good segments", as previously suggested, then the question arises as to how to identify such segments, and more broadly, whether there may exist alternative types of linking that can produce other tour constructions worth considering. We will show it is possible to exploit the reference structure concept to create more powerful processes whose moves include those of the Subpath Ejection Method as a special case.

3.3. *A connection to alternating paths*

There is a link between the preceding Subpath Ejection Method and classical alternating path constructions from graph theory (see, e.g., [1, 14]). Assume the edges of a path are numbered consecutively according to their order of occurrence. An alternating path may be defined as an edge simple path in which even-numbered edges belong to a current solution subgraph, and odd-numbered edges do not. (In typical graph theory applications, the current solution subgraph is one whose edges define a matching or satisfy other degree constrained conditions.)

The purpose of such a path is to identify a transformation of the current solution into another by successively adding and deleting path edges according to whether they are absent or present, respectively, in the current solution. Each edge to be deleted is determined by its adjacency to the preceding edge that is added (at the endpoint to which the path is currently traced) in order to maintain feasibility. In the matching problem context, primal methods use this construction to generate cost improving alternating cycles, while dual methods use it to augment the current solution by reference to successive shortest (minimum cost) alternating paths. (See, for example, [4].)

Although derived from a different perspective, it is not hard to see that the Subpath Ejection Method implicitly generates a type of alternating path construction. This interpretation arises by conceiving the ejection of Step 2 to be broken into a sequence of four operations, first adding the non-tour edge (t, j), followed by deleting the tour edge (j', j), adding the capping edge (j', k''), and finally deleting the tour edge (k, k''). The ordering of this sequence, and the fact that node k becomes the new t, reveals the correspondence with an alternating path. (It is shown in Glover [10] that the Lin–Kernighan heuristic also can be interpreted as a form of alternating path procedure.) The following sections extend this orientation to establish further links between alternating path constructions and the traveling salesman problem.

4. A fundamental stem-and-cycle approach

We now describe an ejection chain process, likewise guided by the stem-and-cycle reference structure, which is able to generate alternating paths not accessible to the Subpath Ejection Method, and at the same time to produce constructions that differ from alternating paths. Rather than seek to eject (and attach) tour segments in conjunction with a capping operation, the process is organized to apply more fundamental ejection steps that preserve the stem-and-cycle structure. In addition, we replace the restrictions based on identifying the critical nodes by a simpler restriction, stipulating only that any edge deleted must not be added back. Each step then consists of adding a single edge and deleting another. We subdivide the rules for maintaining the stem-and-cycle reference structure into those that produce an alternating path and those that do not.

Fundamental stem-and-cycle rules.

Rules to produce an alternating path.

Rule 1: Add an edge (t, j) where j belongs to the cycle. Identify the deleted edge (j, q) to be either of the two edges of the cycle incident at j. Node q becomes the new tip t.

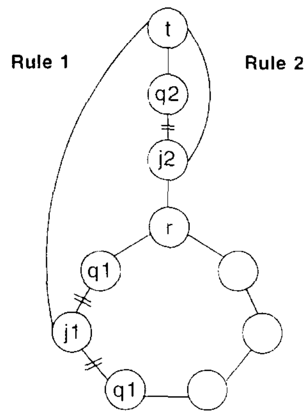
Rule 2: Add an edge (t, j) where j belongs to the stem. Identify the edge deleted (j, q) by requiring q to lie on the portion of the stem between j and t. Node q becomes the new tip t.

Auxiliary rules.

Rule 3: Add (t, j), for an arbitrary node j. Delete an edge (r, q) incident at r, where q is restricted to be a subroot if j is not a cycle node. Node q becomes the new tip t.

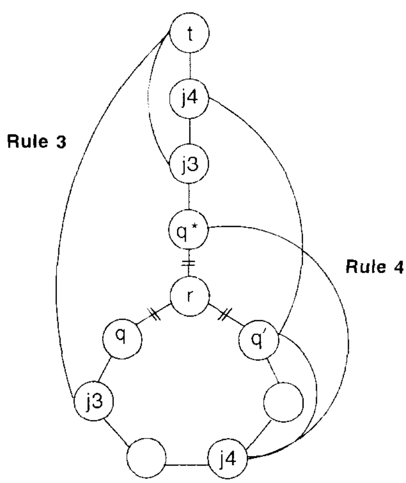
Rule 4: Add (q, j), where q is adjacent to r, and delete the uniquely identified edge (r, q). Node j is an arbitrary node if q is a subroot and otherwise j is a cycle node. Node t remains the tip node.

The application of these rules is illustrated in Figs. 3 and 4. Note that the root never changes its identity. In Rule 1, if j is one of the subroots and q is the root r, the stem-and-cycle reduces to a cycle (with a degenerate stem), causing the root r and new



Rule 1: Add (t,j1), delete either edge (q1,j1)
Rule 2: Add (t,j2), delete edge (q2,j2)

Fig. 3. Stem-and-cycle rules to generate an alternating path.



Rule 3: Add either edge (t,j3), delete any edge meeting r, excluding (r,q) for j3 on the stem.*
Rule 4: Add (q,j4) and delete (r,q*), or add either edge (q′,j4) and delete (r,q′).*

Fig. 4. Auxiliary stem-and-cycle rules.

tip t to coincide. Also, since r belongs both to the stem and the cycle, both Rule 1 and Rule 2 apply when $j = r$, and hence in this case there are three options for identifying the edge (j, q) to delete. (A null move may be possible that adds and drops the same edge.)

Rules 1 and 2 yield an alternating path since j is a node in common with (t, j) and (j, q) and q is identified as the new t. As in the Subpath Ejection Method, such a path may not represent a sequence of feasible tour modifications in the customary sense, although a feasible tour completion always results by the two trial solution alternatives available at each step. Each step of the Subpath Ejection Method can be broken into either two or three steps of applying Rules 1 and 2, though we will later see that the "aggregate move" orientation has merit under certain conditions.

The alternatives by Rules 3 and 4 slightly overlap with those available by Rules 1 and 2. Rule 3 can create both a new tip and a new root, while Rule 4 always creates a new root. Heuristic choice criteria for applying Rules 1–4 (when the steps are not aggregated) can be based on selecting the combination that yields a best trial solution at each step, breaking ties according to the immediate cost of the move (that is, the cost of the added edge minus the cost of the dropped edge). Tie breaking is important since the trial solutions produced by the moves overlap. A more opportunistic (less expensive) alternative is to use the immediate move costs without reference to trial solutions.

The foregoing rules clearly require $O(n)$ effort to apply. The use of a candidate list to limit the number of edges meeting t (or q in Rule 4) does not change the order of this effort, because of the need to differentiate the stem from the cycle, and to keep track of subpaths that change orientation. However, the $O(n)$ effort of updating can be reduced to an $O(1)$ effort, by means of specialized processes which are treated in a sequel [10].

4.1. *Classes of alternating paths and their implications*

While Rules 3 and 4 offer viable choice possibilities using a stem-and-cycle reference structure, we restrict attention to Rules 1 and 2 in this section in order to analyze implications concerning the generation of alternating paths.

By applying Rules 1 and 2 subject to the stipulation that an edge deleted cannot subsequently be added back, we permit constructions that may not be edge simple in a customary sense. That is, by this stipulation, an edge that is added may possibly be deleted later. The classical conception of an alternating path does not harmonize well with this provision, but rather is *static*, where the edges that qualify as available to be deleted are assumed to be those initially present in the subgraph that is transformed. Instead, we require a conception that allows a path to be *dynamic*, so that the process of adding new edges enlarges the set susceptible to being deleted.[1]

Consequently, we seek a system of classification enabling us to treat alternating paths from such a perspective. For this purpose, we begin by defining an alternating path to be *add simple* (*delete simple*) if no added (deleted) edge appears more than once in the path. Note a path that is both add simple and delete simple is not necessarily

[1] This notion may be viewed as complementary to the ideas of de Werra and Roberts [2], who give a way to generalize alternating paths in the context of chain packing problems.

edge simple in a static sense, since such a path allows the same edge to be both added and deleted.

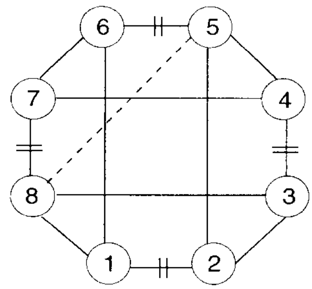
An additional level of differentiation is required to encompass the restriction we have imposed on generating alternating paths. Specifically, we require the notion of a *conditionally simple* alternating path, in which the presence of one class of edges depends on that of another. Define a path to be *delete\add simple* if a deleted edge does not subsequently reappear as an added edge, and to be *add\delete simple* if an added edge does not subsequently reappear as a deleted edge. (By the natural analogs of these definitions, it can be seen that the classes of add\add simple paths and delete\delete simple paths are the same as those of add simple and delete simple paths, respectively. In addition, add\delete simple paths and delete\add simple paths are both instances of paths that are simultaneously add simple and delete simple.) The type of alternating path produced by the restriction we apply with Rules 1 and 2 thus constitutes what we have called a delete\add simple path. Our analysis will be concerned with identifying special properties about the nature and existence of such paths in connection with the traveling salesman problem.

Illustration of a dynamic (delete\add simple) alternating path

A simple example shows the necessity of considering alternating paths more general than customary in order to transform one traveling salesman tour into another. The example is a familiar one known to provide an instance where the Lin–Kernighan construction fails to transform one tour into another, and can be expressed by reference to a graph consisting of eight nodes, shown in Fig. 5.

Starting from the tour that visits the eight nodes in numerical order, the goal is to reach the tour that visits the nodes in the order 1, 6, 7, 4, 5, 2, 3, 8, 1. This tour deletes edges (1, 2), (3, 4), (5, 6), (7, 8), and adds edges (1, 6), (2, 5), (3, 8), (4, 7). It is clear the indicated transformation does not represent a connected alternating path, but rather two disjoint (and piecewise infeasible) alternating paths. (Evidently, the symmetric difference between two tours always can be expressed as a collection of alternating paths (cycles) whose maximum components are node disjoint. Results bearing on this appear in Section 8.) Nevertheless, there does exist a delete\add simple path that transforms the first tour into the second, which makes use of the dotted edge in Fig. 5. We demonstrate this outcome by showing in addition that such a path can be obtained by the stem-and-cycle approach following Rules 1 and 2.

Starting from node 1 as the root r (and hence also the initial tip t), the alternating sequence of edges added and dropped by applying Rules 1 and 2 are as follows: (1, 6), (6, 5), (5, 8), (8, 7), (7, 4), (4, 3), (3, 8), (8, 5), (5, 2), (2, 1). At each step, the second node of the odd-numbered (added) edges represents the current node j in Rules 1 and 2, while the second node of the even-numbered (deleted) edges represents the current node q, which becomes node t at the next step. The last pair of edges added and dropped may also be viewed as the outcome of selecting the trial solution associated with subroot 2 at the next to last step. (The indicated transformation is not the only one that can be obtained by this process.)



Dotted edge is second added, fourth deleted

Fig. 5. A delete\add simple alternating path transformation using a stem-and-cycle construction.

4.2. Delete\add simple paths and the stem-and-cycle structure

Clearly no alternating path exists to transform the first tour of the preceding example into the second except by adding an edge that must subsequently be deleted. At a more general level, the example prompts the question of whether a given tour may be transformed into any other by an alternating path that is delete\add simple.

For this question to be meaningful, we assume either that the graph is dense or that artificial edges may be added as needed (which will also be removed during the path construction). The illustrated path in the example of Fig. 5 also exhibits another property we regard to be significant. After adding the edge (5, 8) that subsequently becomes deleted, the next added edge is one that is not deleted. We call a delete\add simple path with this characteristic, i.e., in which at least one of any two successively added arcs is never subsequently deleted, a *first order* delete\add simple path. Results proved in [10] establish that such an alternating path can transform any tour into any other, and more importantly (from our perspective), that the transformation can always be produced by the stem-and-cycle approach using Rules 1 and 2. A useful implication is that an optimal tour is always potentially accessible at each execution of the stem-and-cycle approach. This outcome may be viewed as a *connectivity* result in the space defined by these special types of alternating paths.

Moreover, the rules described for the symmetric case can be modified in a natural manner to handle asymmetric traveling salesman problems. This is a feature not shared by some popular approaches, such as 2-opt and the Lin–Kernighan procedure, which entail reversals of subpaths in order to achieve such an extension.

Next we examine how these ideas can be exploited in a parallel processing framework, which gives an implicit ability to generate structures more varied than by a serial approach.

5. Parallel processing and stem-and-cycle structures

The issue of applying the stem-and-cycle reference structure to solve traveling salesman problems by parallel processing introduces concerns that are particularly relevant to solving large problems. We show it is possible to organize separate stem-and-cycle processes into a method that creates a graph with multiple cycles and stems, not necessarily connected, while operating only with the rules previously described.

Our approach is based on the simple strategy of creating *boundary nodes* that subdivide a selected starting tour into subpaths. Each parallel process then is applied to the nodes of a given subpath (see, e.g., [6, 20]). Our goal is to implement an ejection chain approach on the subgraph induced by the selected subpath nodes, in such a way that the stem-and-cycle structures of the separate processes can always be concatenated to create a single feasible tour.

Let $N(u, v)$ denote the set of nodes of a given starting subpath, initiated by the boundary node u and terminated by the boundary node v. By applying the stem-and-cycle reference structure to this set of nodes, the rules for deleting edges will create stems and cycles that implicitly must be routed through nodes outside this set (in order to maintain an appropriate structure). The effect of operating in parallel on other sets of nodes of the graph, thereby introducing other stem-and-cycle configurations external to $N(u, v)$, creates a "divided" stem-and-cycle structure that destroys the simple external path routing on which the rules for altering the structure relative to $N(u, v)$ presumably rely. Some device must be employed to assure that the diverse components of the divided structure are susceptible to relinking. Fortunately, this turns out to be easy to do, as noted in the following observation.

Remark. To operate on the subpaths in parallel, create an initial subtour for each pair of successive boundary nodes u and v, consisting of the starting path from u to v and a single artificial edge (u, v). Then apply the stem-and-cycle preservation rules to each such subtour and the subgraph induced by $N(u, v)$, subject to the restriction that edge (u, v) is never deleted. The trial solutions for the separate stem-and-cycle structures create a feasible tour over all nodes upon removing the artificial edges.

The foregoing remark is immediately justified by the fact that each trial solution must generate a feasible tour over $N(u, v)$, and this tour will constitute a path from u to v upon eliminating the artificial edge. The union of the paths is evidently a feasible tour.

An example of a divided stem-and-cycle structure generated by the approach of the Remark is shown in Fig. 6. The starting structure represents one obtained after a first step that adds and deletes an edge in each subtour to create a non-degenerate stem-and-cycle. The three square nodes of the figure identify the selected boundary nodes and the artificial edges (not shown) thus connect each successive pair of these nodes. The final stem-and-cycle structure shown in Fig. 6 is obtained after two

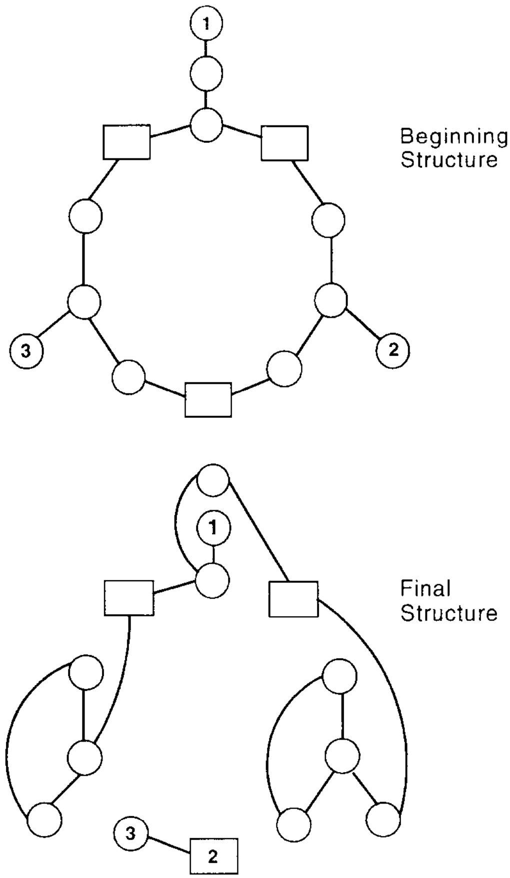


Fig. 6. Divided stem-and-cycle structure for parallel processing.

iterations applying Rules 1 and 2 to each subtour, subject to not deleting artificial edges. The stem tips for each of the three initial and final component configurations are indicated by the nodes numbered 1, 2 and 3.

Either of the two trial solutions for each component can be linked with remaining trial solutions to create a feasible tour. In practice, the best trial solution obtained by parallel processing over each component will be retained to construct the desired composite tour. Then another set of boundary nodes is selected to create a different division into subpaths, and the process repeats.

We now show how the simple concept of the preceding Remark can be usefully broadened.

Extended Remark. Let $u_1, v_1, u_2, v_2, \ldots, u_k, v_k$, be a succession of consecutive boundary nodes. Create an initial subtour consisting of the starting subpaths over each of the sets $N(u_i, v_i), i = 1, \ldots, k$, together with linking artificial edges $(v_i, u_{i+1}), i = 1, \ldots, k$, where $u_{k+1} = u_1$. Then apply the stem-and-cycle rules over the composite subgraph induced by the union of the $N(u_j, v_i)$ node sets, subject to the restriction that no artificial edge is deleted, and independently apply the approach over each separate subgraph induced by the node sets $N(v_i, u_{i+1})$, by the rule of the preceding Remark. The trail solutions for each of these subgraphs can then be linked to create a feasible tour by deleting the artificial edges.

The significance of the Extended Remark is to permit a strategy of identifying the sets $N(u_i, v_i)$ with the goal of incorporating nodes which possibly should be redistributed among these sets. In particular, the application of the Extended Remark makes it possible for the final trial solution subpaths (from each u_i to each v_i) to incorporate nodes other than those contained in the sets $N(u_i, v_i)$, as initially identified. Further, these subpaths may become linked to the other subpaths in a sequence different from that indicated by their indexes. (The added flexibility of this second condition prevents the simultaneous creation of another subgraph induced by the union of selected remaining subpaths, since the final subtours obtained from these "aggregate subgraphs" may arrange their components in incompatible sequences.)

Finally, the Extended Remark can be applied recursively. That is, each node subset $N(v_i, u_{i+1})$ and its associated starting subpath can be subdivided by the rules of the Extended Remark exactly as if it represented the node set of the complete problem. In this way, when boundary nodes are reselected in a repetitive application of the approach, it is possible to allow interactions among non-adjacent subpaths on the basis of varying criteria. We note these observations apply not only to the application of the stem-and-cycle approach, but also to any procedure capable of recovering a feasible trial solution at each step (over the subgraph considered).

To exploit parallel processing opportunities as fully as possible in this setting, it is useful to supplement the possibilities made available by the Extended Remark. Ideally, we would seek to encompass interactions across non-adjacent subpaths that do not depend solely on the recursive identification of boundary points to qualify as the pairs (u_i, v_i). This goal can be pursued by an ejection chain strategy using the constructions of [8]. A method using these constructions can easily control subdivisions of a tour so that component subpaths will not become resequenced relative to each other. Thus, in a parallel processing environment, such an approach may provide a useful companion to ejection chain approaches based on the foregoing ideas.

6. Doubly rooted reference structures

We identify a reference structure in this section that not only permits more direct trajectories between tours, but provides moves beyond those available by the combination of the preceding approaches. This structure requires only slightly more effort to manage than the stem-and-cycle structure, still involving O(n) computation at each execution. The structure may be conceived as arising from a stem-and-cycle by introducing a single additional edge (t, j), which connects stem tip t to an arbitrary node j (without simultaneously deleting an edge). Node j thus identifies a second root (possibly coinciding with the first root), and accordingly we call this construction a *doubly rooted* reference structure. When the two roots are distinct, each meets exactly three edges of the structure, and when they coincide, they meet four edges of the structure. All other nodes meet exactly two edges.

The doubly rooted structure has two forms: a *tricycle* in which the two roots are connected by the three paths, thereby generating three cycles (two that share the "inner" path between the roots, and one that does not); and a *bicycle* in which the roots are connected by a single path, joining two cycles. (A degenerate bicycle is produced when the roots correspond.) These forms are illustrated in Fig. 7.

The doubly rooted structure contains no explicit tips, but contains up to six *implicit* tips identified by deleting edges (one at a time) that meet the roots. Thus, the implicit tips coincide with the subroots, which we define to constitute the nodes adjacent to the roots. Subroots are divided into two classes, *cycle subroots* and *non-cycle subroots*, where the latter are those that lie on the path between the two roots of a bicycle. (A tricycle and a degenerate bicycle contain only cycle subroots.) The roots can also be

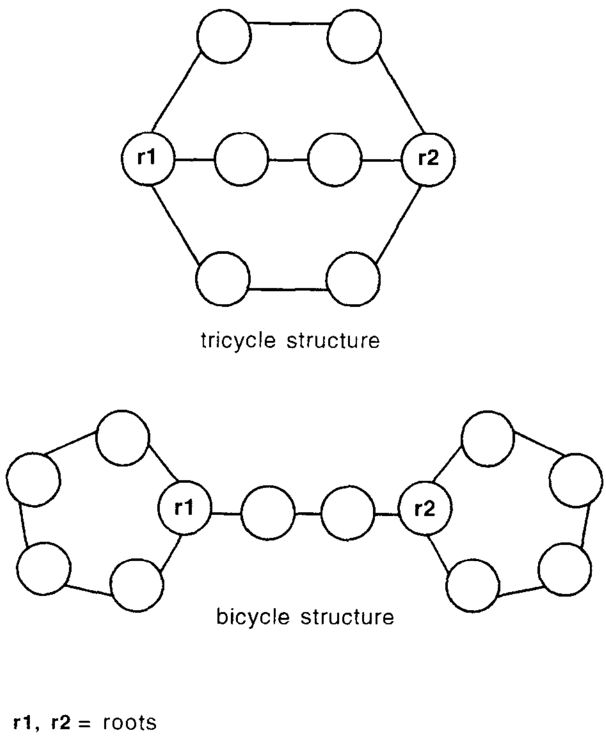


Fig. 7. Doubly rooted stem-and-cycle structures.

subroots if they join by an edge. In this special case, a root r of a bicycle that is also a subroot must by definition be a non-cycle subroot. Although it lies on a cycle, the cycle is not shared with the root to which it is adjacent.

A stem-and-cycle structure results by deleting any edge (r, s) such that r is a root and s is a cycle subroot. Node s becomes the tip, while the (root) node that remains with three incident edges becomes the stem-and-cycle root. The trial solutions available to the doubly rooted structure are thus the union of the trial solutions available to these component stem-and-cycle structures. In fact, the trial solutions that result by transforming a cycle subroot s into a tip, for each such s associated with a given root r, are the same as the trial solutions similarly produced from the subroots of the other root, and hence attention can be restricted to only one of the two sets of subroots for this purpose. (Each cycle subroot of a given root produces two trial solutions.) The enriched pool of such trial solutions, together with an enriched set of moves for transitioning from one reference structure to the next, provide the potential advantage of the doubly rooted structure. The rules to transition between structures are as follows.

Rules for the doubly rooted structure.

Rule 1-DR: Select a cycle subroot s and an associated root r. Add an arbitrary new edge (s, j) (not in the current structure) and delete the edge (s, r). After the step, j becomes a root (and r is no longer a root unless the two roots coincided before the step).

Rule 2-DR: Select a non-cycle subroot s and an associated root r. Add a new edge (s, j) such that j lies on the cycle in common with r, and delete (s, r). Node j becomes a new root (and r is no longer a root).

These two rules, although simple to describe, encompass all alternatives available to the Subpath Ejection Method and all possibilities contained in Rules 1–4 of the stem-and-cycle approach, as applied to each of the component stem-and-cycle structures implicit within the doubly rooted structure. This outcome is somewhat counterintuitive, since the stem-and-cycle rules appear to offer a broader range of options, allowing for the deletion of edges not specified in Rules 1-DR and 2-DR.

The broader purview of Rules 1-DR and 2-DR becomes understandable by considering how they achieve the effect of Rule 1 for the stem-and-cycle structure, which adds an edge (t, j) from the tip t to the cycle, and drops an adjacent cycle edge (q, j). The corresponding doubly rooted structure contains an edge that joins t to a root r', which makes t a cycle subroot of r'. Assume the same edge (t, j) is added to this augmented structure. No corresponding deletion of an edge (q, j) occurs by Rule 1-DR or 2-DR. The fact that j becomes a new root means that q corresponds a new subroot. (If $j = r'$ the root and subroot are unchanged.) It thus can be seen that the trial solutions available to the stem-and-cycle structure after applying Rule 1 (and dropping (q, j)) are a subset of those available to the doubly rooted structure after applying Rule 1-DR or 2-DR (and making j a new root). Moreover, the doubly rooted structure has the option now to execute a move that drops (q, j), which implicitly corresponds to choosing q as the new tip t in the stem-and-cycle approach. However, we are not limited to the options that result by specifying q to be the new tip, and hence an expanded set of move possibilities is available.

Rules 1-DR and 2-DR of course do not in general create a connected path sequence, and thus go beyond alternating path structures, though again we will be interested in such constructions as an important special case. To prevent a return to an alternative available on the preceding step, an edge deleted may be prevented from immediately being added back (or, more broadly, prescriptions of the form used with tabu search can be applied).

6.1. The doubly rooted structure for the asymmetric problem

The form of the doubly rooted structure for the asymmetric problem is a direct analog of the one for the symmetric problem. The asymmetric structure gives rise to three directed cycles in the case of a tricycle, and to two directed cycles, joined by a directed path, in the case of a bicycle. (The directed path may have no arcs if the roots coincide.) When the roots are distinct, one root has two arcs entering and one arc leaving, while the other root has two arcs leaving and one arc entering. Each root has two subroots (instead of three), which lie on the two arcs that enter or the two arcs that leave the root. In the degenerate case, as before, each of the four nodes adjacent to the root is a subroot. The distinction between cycle subroots and non-cycle subroots remains unchanged.

By these conventions, the rules for the directed case of the doubly rooted structure are exactly the same as Rules 1-DR and 2-DR, except that the added and deleted arcs must be directed the same relative to the subroot s (i.e., the added and deleted pair is either (s, j) and (s, r) or (j, s) and (r, s)).

6.2. Improved alternating paths

The ability of the doubly rooted structure to generate more effective alternating path sequences can be established rigorously as follows.

Theorem 1. *There exists a first order delete\add simple alternating path, starting from an arbitrary node r as a first root, and obtainable by the doubly rooted structure using Rules* 1-DR *and* 2-DR, *that will transform an initial directed tour into another directed tour by adding less than $2m$ arcs, where m is the number of arcs in the second tour not in the first.*

Proof. We begin by adding a non-tour arc (r, r^*) to the initial tour to create a doubly rooted structure with roots r and r^*. Since the doubly rooted structure always contains one more arc than a tour, we seek a transformation that yields the second tour plus a single arc (which is dropped in a final degenerate trial solution). As before, arcs belonging to the second tour will be classified white and others black, and we identify a linked succession of steps that assures the path is alternating.

In the worst case, the first non-tour arc added is compelled to be black (where the tour arc out of node r is white). Then choose r^* so that the unique arc (k, r^*) into r^* also is black. (If instead (r, r^*) is white, then (k, r^*) automatically is black.) In general, we use such an approach to assure each step of adding an arc likewise creates a root r^* that is met by an associated arc (k, r^*) which is black. At this point k is a subroot s of r^*, since the doubly rooted structure has two arcs entering r^*. The next step therefore will be to drop the black arc (s, r^*). If this deletion creates a tour (corresponding to a degenerate trial solution), and if all arcs are white, the process is completed. Otherwise, we add a new arc (s, j) by either Rule 1-DR or 2-DR.

Assume first s is a cycle subroot (as it must be on the first step after adding (r, r^*)). Then Rule 1-DR applies, permitting j to be an arbitrary node. We therefore can select (s, j) to be white, provided either that s was not a root node before deleting (s, r^*), or else that there is no white arc out of this root. Then, upon identifying j as the new r^*, we are ready to begin again under the same conditions that initiated the preceding step (where (k, r^*) is black, etc.).

The only way to break a sequence of add-drop steps where every added arc is white (and every dropped arc is black) is therefore reduced to two cases: (1) encountering a non-cycle subroot s that renders Rule 1-DR inapplicable; or (2) encountering a cycle subroot s that also is a root with a white arc out of it. First suppose (1) applies. Then the node j of the added arc (s, j) must lie on the cycle containing the root r^* (where this cycle is disconnected from the rest of the structure by deleting (s, r^*)). If the white arc out of node s does not lie on this cycle, the cycle must contain at least one black arc, which we denote by (k, j), thus identifying node j chosen for the endpoint of the added arc (s, j). Since j becomes the new r^*, we fulfill the previous claim that the arc (k, r^*) is always black. Further, node k must in fact be a cycle subroot, since k and the new r^* lie on the same cycle identified by reference to the previous r^*. Consequently, Rule 1-DR is applicable when k becomes s on the following step, and we conclude s cannot be a root. This assures that the next arc added is white, as desired.

Next suppose case (2) applies. Node s can be both a cycle subroot and a root that already initiates a white arc only if the deletion of (s, r^*) reduces the structure to a cycle. Thus, if the cycle is not the desired tour, and any black arc remains, we can select such an arc as (k, j), and by the argument just given we conclude a white arc can be added on the step after adding a black arc. The alternating path therefore is a first order path. Fewer than $2m$ arcs are added, due to the first order condition and the fact that at least two white arcs must be added after the last black arc is added.

Finally, we must show the path is delete\add simple. Suppose on the contrary a black arc (k, j) is deleted and then subsequently added. First, we argue that (k, j) cannot be deleted in a step that also adds a white arc (s, j). This would prevent (k, j) from being added later by the construction previously indicated, which permits a black arc to be added to meet a node j only if there is exactly one arc into j and this arc is black. The presence of a white arc (s, j) renders this impossible, and hence (s, j) must be black. We have shown a new white arc (k, j') is added on the next step, and at this point no black arc exists out of node k. In order to add any new arc (k, q) out of node k (and in particular (k, j)), it is necessary first to drop some arc (k, h) out of k. Moreover (k, h) must be black since the process never drops a white arc. But this is impossible, since no black arc currently exists out of node k, and none can be added (unless one exists already to be dropped). We therefore conclude the path is delete\add simple. □

Corollary to Theorem 1. *The statement of Theorem* 3 *also is valid for the symmetric TSP* (*upon stipulating that tours are undirected*).

Proof. The result follows by an argument that parallels the proof for the directed case. □

Theorem 1 is a stronger result than the theorems for stem-and-cycle reference structures, which raises the question of whether this outcome may have practical as well as theoretical significance. From an applied standpoint, the potential value of the doubly rooted reference structure rests on the tradeoff in effort required to manage it and on identifying choice rules that can capitalize on its expanded range of options. It is shown in [10] that the doubly rooted structure offers particular benefits in designing a method to create a combinatorial leverage effect.

7. Stem-and-multicycle reference structures

The relationship between the stem-and-cycle structure and the doubly rooted structure suggests that a further advanced reference structure is likely to encompass an additional number of edges (or arcs). However, we will show that a reference structure satisfying the desired conditions exists which contains only the same number of edges as in a tour. This structure, which we call the *stem-and-multicycle* structure, is a spanning subgraph whose components include a stem-and-cycle, denoted S–C, and a collection of cycles, denoted $C(h)$, $h \in H$ (where H may be empty). The components of the stem-and-multicycle are pairwise node disjoint. An illustration of this structure appears in Fig. 8.

Although the organization and transition rules of the stem-and-multicycle structure are more complex than those of the reference structures previously discussed, they are not difficult to manage. Each step for modifying the structure, as before, is a simple add-drop operation. The effort required to perform each step remains $O(n)$.

We first describe the rules for transitioning from one stem-and-multicycle structure to another, and then indicate how these rules may be integrated with the generation of trial solutions.

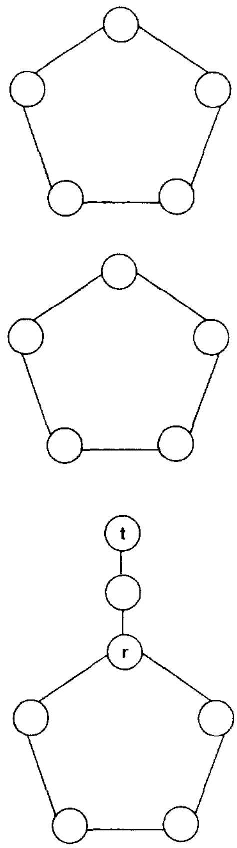


Fig. 8. Stem-and-multicycle structure.

7.1. Transition rules for the stem-and-multicycle reference structure

For convenience, we let SMC denote the full stem-and-multicycle structure, and let S and C respectively denote the stem and the cycle of the S–C component. The rules to modify SMC consist exactly of Rules 1 and 2 for the ordinary stem-and-cycle, plus two others. We identify these rules as follows.

Stem-and-multicycle rules.

Rule 1*: Apply Rule 1 to S–C.

Rule 2*: Apply Rule 2 to S–C.

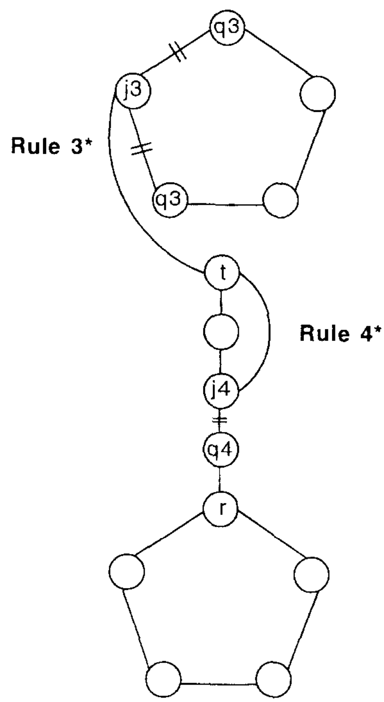
Rule 3*: Add edge (t, j), where t is the tip of S, and node j lies on some cycle $C(h)$, $h \in H$. Delete either cycle edge (q, j) meeting j. Node q becomes the new tip t of S (which is now augmented), and $C(h)$ is removed from the collection of cycles (by deleting h from H).

Rule 4*: Add edge (t, j), where t is the tip of S and node j lies on S, with $j \neq r$. Delete edge (q, j), where q is the stem node adjacent to j that lies closer to r. Edge (t, j) together with the segment $(j, \ldots, t)$ of the stem becomes a new cycle $C(h)$, $h \in H$, and node q becomes the new tip of S (which is now reduced).

Special provision. If S becomes degenerate as a result of Rule 1* or 4* (hence causing the cycle C to compose all of S–C), then a new root r and tip t may be selected (with $r = t$).

The preceding special provision can be accompanied by changing the designation of the cycle identified as S–C, if desired. Also, after executing the provision, the method optionally may restrict attention to applying Rule 3* on the succeeding step, if H is not empty. Rules 3* and 4*, which provide the elements of this approach that are new, are illustrated in Fig. 9.

We establish that the foregoing rules indeed suffice to transform one tour into another without superfluous moves. In fact, we give a stronger result which shows that the stem-and-multicycle reference structure in a sense is the "correct" structure for



Rule 3*: Add (t,j3), delete either edge (q3,j3)
Rule 4*: Add (t,j4), delete edge (q4,j4)

Fig. 9. Additional rules of the stem-and-multicycle reference structure.

transforming one tour into another, for the goal of creating a collection of alternating paths (cycles) that yield a partition of the symmetric difference of the tours. More precisely, by the preceding rules, the structure makes it possible to produce any such collection that is capable of transforming the first tour into the second. We state this result as follows, where alternating cycles are understood to be defined relative to the tours considered.

Theorem 2. *Let T_1 and T_2 be two distinct tours, and let C_i, $i \in I$, be a collection of edge simple alternating cycles, pairwise edge disjoint, which exactly describe the symmetric difference between T_1 and T_2 (i.e., $T_1 - T_2 = T_1 \cap C$ and $T_2 - T_1 = T_2 \cap C$, for $C = \bigcup\{C_i : i \in I\}$. Then Rules 1* – 4* can be applied to generate precisely each cycle of the collection C_i, $i \in I$, starting from T_1. The stem-and-multicycle structure produced at each step corresponds to the transformation of T_1 produced by the current subcollection of alternating paths, and the final structure corresponds to T_2.*

Proof. Designate edges of T_2 to be white and others black. We begin with SMC as the tour T_1, and select r (with $r = t$) to be any node that meets a white edge not in SMC. This edge, (t, j), belongs to some alternating cycle C_i, and an adjacent black edge (q, j) of C_i must exist that also lies on T_1, hence SMC. Rule 1 permits this edge to be deleted, transforming SMC into the S–C structure where S is non-degenerate. SMC contains all edges of T_1 except those already accounted for as part of C_i, and on a general step contains all edges except those that are elements of paths previously generated or in the process of being generated. Further, the white edges that have not yet become a part of SMC are exactly those edges in cycles that are not yet accounted for, and this also holds on a general step. Consequently, upon designating the new tip t to be q, we are assured the next white edge (t, j) of C_i is not in SMC, and more broadly, whenever S is non-degenerate (and hence t has degree 1 in SMC), there must exist a white edge (t, j) of a current C_i being generated that likewise is not in SMC. (The cycle cannot be completed until dropping an edge results in $t = r$, creating a degenerate stem.) Hence we now select this white (t, j) to add to SMC. At least one of the edges (q, j) of SMC must be the next black edge of C_i. Rules 1*–4* permit this edge to be deleted, regardless of its identity, by the following correspondences: (1) Rule 1* governs if (q, j) lies on C; (2) Rule 2* governs if (q, j) lies on S and q is on the path from j to t; (3) Rule 3* governs if (q, j) lies on a cycle $C(h)$; (4) Rule 4* governs if (q, j) lies on S and q is on the path from j to r. In each case, a new SMC is created that preserves the conditions previously identified and permits Rules 1*–4* to be applied again. Upon encountering a degenerate S, an alternating cycle is produced. This cycle may not be all of C_i, since C_i is not required to be node simple. In this case, we continue exactly as before, without changing r. But if C_i is fully generated, and if SMC does not yet correspond to T_2, then there must be some node r of C that meets a white edge not in SMC. (If H is not empty, we further can find such an edge that joins C to some other cycle of SMC.) The argument now repeats, permitting the generation of the (new) cycle C_i that contains this white edge meeting r. Eventually, all white edges will be brought into SMC, thus assuring that all cycles are generated, and causing SMC to correspond to T_2. □

The foregoing proof also can be used as a constructive argument to show the existence of a collection C_i, $i \in I$ of the form indicated (allowing some simplification of the proof for this purpose).

The stem-and-multicycle structures for the asymmetric TSP

For the asymmetric problem, we assume all cycles are directed cycles, and S is directed from r to t. The rules applicable to this problem are as follows.

Directed stem-and-multicycle rules.

Rule 1A*: Apply Rule 1A to SC.

Rule 2A*: (Nonexistent).

Rule 3A*: Replace "edge" by "arc" in Rule 3*, and delete the unique cycle arc (q, j) meeting j.

Rule 4A*: Replace "edge" by "arc" in Rule 4*.

Corollary to Theorem 2. *Theorem* 4 *is valid for the asymmetric problem by replacing Rules* 1*–4* *with Rules* 1A*–4A*. *In addition, the application of these rules identifies the collection of alternating cycles* $C_i, i \in I$ *to be uniquely determined, given* T_1 *and* T_2.

Proof. The argument takes a form analogous to the proof of Theorem 2. The unique identity of the collection $C_i, i \in I$ follows from the fact that as long as S is non-degenerate, the identity of each (t, j) and (q, j) is uniquely determined, and the generation of the current C_i is completed as soon as S becomes degenerate. □

7.2. Trial solutions for the stem-and-multicycle structure

To characterize the form of trial solutions, and also to update the stem-and-multicycle reference structure, we stipulate the use of linked lists identifying the predecessor and successor of each node in the cycles $C(h)$, $h \in H$, and in S–C. (When S is non-degenerate, the tip t has no successor and r has two successors, one on S and one on C.) All observations are expressed in terms of the symmetric problem, but can be applied in an evident manner to the asymmetric problem also.

Assume H consists of positive indexes only, and each node i of a cycle $C(h)$ is given a label $\pi(i) = h$. Also, label each node i of S, except r, by $\pi(i) = 0$ and each node i of C by $\pi(i) = -1$. Changes in the labels that result by applying Rules 1*–4* can easily be carried out in $O(n)$ time. Associated trial solutions are generated as follows.

For each cycle $C(h)$, we identify a trial edge $(u(h), v(h))$, whose removal creates a *partial stem* $S(h)$ with *tip* $u(h)$ and *tail* $v(h)$. The cycles, and hence partial stems, are ordered in a strict sequence, each with a predecessor and successor cycle (except that the first lacks a predecessor and the last lacks a successor). Thus we connect the tip $u(h)$ and tail $v(h)$ of $S(h)$ by edges $(u(h), v)$, and $(u, v(h))$ to the tail v of the successor partial stem and the tip u of the predecessor partial stem. If $S(h)$ is the first partial stem, the node u of $(u, v(h))$ is in fact t of the stem S, thus creating a linking that establishes a complete stem S^* of a trial stem-and-cycle structure. The tip $u(h)$ of the last partial stem is also the tip t^* of S^*. An illustration of such a linking appears in Fig. 10. This trial stem-and-cycle structure provides the source of trial solutions by the customary rules.

Once the trial edges are chosen and a linking is established, updates to the linking and the identification of new trial solutions for each tentative move examined by Rules 1*–4* can be achieved in constant time. In the case where a move would destroy a cycle, a sequential ordering of the remaining cycles (partial stems) is preserved by relinking the predecessor and successor of the cycle removed (allowing a simplified relinking if this cycle is the first or the last in the sequence). If instead the move would create a cycle, this cycle becomes the new "first" $C(h), h \in H$. (The effort of the preceding updates is independent of the size of H once H contains two or more elements.)

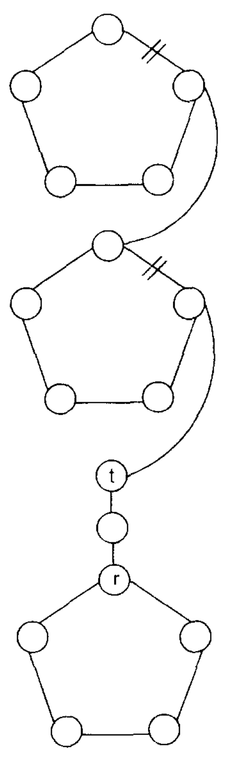


Fig. 10. Linked partial stems (to create trial solutions).

The precise rules for changing the links defining the trial solutions, except those already established for linking the tip t^* to the associated subroots of r, may be stated as follows.

Changes to S^*.

After applying Rule 1* *or Rule* 2*. Drop $(t, v(1))$ and add $(q, v(1))$. Node q becomes the new t and t^* is unchanged.

After applying Rule 3*. Let $C(h')$ and $C(h'')$ respectively denote the cycles that precede and follow $C(h)$, where $C(h)$ contains the deleted edge (q, j).

Case 1: $C(h')$ *and* $C(h'')$ *both exist.* Drop $(t, v(1)), (u(h), v(h''))$ and $(u(h'), v(h))$. Add $(q, v(1)), (u(h), v(h))$, and $(u(h'), v(h''))$.

Case 2: $C(h')$ *exists but* $C(h'')$ *does not.* Drop $(t, v(1))$ and $(u(h'), v(h))$. Add $(q, v(1))$ and $(u(h), v(h))$. $C(h')$ becomes the new "last" cycle and $u(h')$ becomes the new t^*.

Case 3: $C(h'')$ *exists but* $C(h')$ *does not.* Drop $(t, v(1))$ and $(u(h), v(h''))$. Add $(q, v(h''))$ and $(u(h), v(h))$.

Case 4: $C(h')$ and $C(h'')$ *both do not exist.* Drop $(t, v(1))$ and add $(u(h), v(h))$. Node q becomes the new t^* (and H becomes empty).

After applying Rule 4*. No change in S^* occurs. (Edge (t, j) becomes the trial edge $(u(1), v(1))$ for the newly created cycle $C(1)$.)

We express these as changes to S^*, which occur in addition to the changes of adding (t, j) and dropping (q, j) after applying each of the Rules 1*–4*. We assume H is non-empty for Rules 1*–3*, for otherwise the changes are those already stipulated for a simple stem-and-cycle. Also, we use the convention that $C(1)$ denotes the "first" $C(h)$, $h \in H$, hence identifying $(u(1), v(1))$ as the trial edge associated with this cycle.

Note in the cases associated with Rule 3*, the changes should be interpreted as occurring before the addition of (t, j) and deletion of (q, j). Thus, the addition of $(u(h), v(h))$, which restores $S(h)$ to $C(h)$, may be offset by the deletion of (q, j), if the two edges are the same. Similarly, the deletion of $(t, v(1))$ may be offset by the addition of (t, j) in Case 4, if these edges are the same. The fact that S^* does not change after applying Rule 4* underscores the importance of evaluating moves by reference to cost changes other than (or in addition to) those produced by trial solutions. For example, trial solutions may be used as a secondary evaluation criterion except where one of particularly high quality is produced.

The identification of (t, j) as the trial edge $(u(1), v(1))$ after applying Rule 4* of course may not be the best choice, and an option that can be quickly tested is to consider the more costly of the two edges adjacent to (t, j) on $C(1)$ as a candidate for $(u(1), v(1))$. Such options normally will be restricted during the examination of potential moves so that each move can be evaluated in constant time. However, once a move is selected and executed (or a preferred of moves are isolated for more extensive evaluation), superior choices for the trial edges may be considered. The next section examines this issue.

7.3. *Identifying trial edges and relinking partial stems*

We describe a straightforward method to identify trial edges for the goal of determining improved trial solutions, constituting a local improvement process that can be applied in O(n) time (or less). For this, let $H^* = H \cup \{0, -1\}$ and consider two additional cycles, $C(0)$ and $C(-1)$. $C(-1)$ is just C (recalling that each node i of C is labeled $\pi(i) = -1$). We create $C(0)$ by first identifying the partial stem $S(0)$ that spans the nodes i labeled $\pi(i) = 0$ (all nodes of S except r), and then adding an edge joining the endpoints of $S(0)$ to complete the cycle. For this we assume S is non-degenerate (else $S(0)$ has no nodes), and denote the endpoints of $S(0)$ by $u(0)$ and $v(0)$, where $u(0) = t$ and $v(0)$ is the node that links $S(0)$ to C by the edge $(r, v(0))$.

The cycle $C(0)$, created by adding $(u(0), v(0))$ to $S(0)$, may be degenerate if $S(0)$ has only one node, and also (in the symmetric problem) if $S(0)$ has two nodes, since then the added edge $(u(0), v(0))$ duplicates the single edge of $S(0)$.

We identify the trial edge $(u(-1), v(-1))$ of $C(-1)$ by stipulating that $u(-1) = r$ and $v(-1)$ is the subroot s of C that yields a preferred trial solution (upon deleting (r, s)). H^* is ordered just as H is ordered, taking -1 and 0 to be the first two elements of H^*, followed by the first element of H (if H is non-empty). H^* further is treated as cyclic, where the last element of H^* precedes the first. (In special case where S is degenerate and $S(0)$ does not exist, we remove the index 0 from H^*.) Thus, the linking of partial stems $S(h)$ by adding $(u(h), v(h''))$, where h'' follows h, for each $h \in H^*$, corresponds exactly to the trial solution previously specified for SMC. This holds true also for the case where $S(0)$ contains only one or two nodes (and $C(0)$ is degenerate).

Given this framework, the local improvement procedure for determining a better trial solution seeks an identity for $(u(h), v(h))$ in each $C(h)$, so that the linking of the resulting partial stems $S(h)$ yields a least cost tour. We observe that $u(0)$ and $v(0)$ must be held invariant if $S(0)$ contains no more than two nodes, and hence $C(0)$ is excluded from the process in this circumstance.

Local improvement of trial edges.

Initialization. Let L be a list of all $h \in H^*$ such that $C(h)$, or its predecessor or successor cycle in H^*, has changed since the last execution of this method.

Step 1: Select and remove some h from L, and identify the predecessor h' and successor h'' of h in H^*.

Step 2: Select $(u(h), v(h))$ to be an edge of $C(h)$ that minimizes $c(u(h), v(h'')) + c(u(h'), v(h)) - c(u(h), v(h))$. If $u(h)$ or $v(h)$ changes its identity, add h' and h'' to L. Then return to Step 1 unless L is empty.

The identity of $u(h)$ and $v(h)$ in the edge $(u(h), v(h))$ implicitly "orders" these nodes. That is, if $u(h)$ and $v(h)$ are interchanged in the minimization criterion of Step 2, a different outcome results. Thus each (i, j) of $C(h)$ is examined twice, once to check for $i = u(h)$ and $j = v(h)$, and once to check for $i = v(h)$ and $j = u(h)$. This procedure can alter the stem S if it selects $u(0)$ to be different from t, and also can alter the root of C if it selects $u(-1)$ to be different from r. Since the number of edges in all cycles on the list L is at most n, the method requires at most $O(n)$ effort between successive improvements, or before terminating after the last improvement.

7.4. A global method for identifying best trial edges

We now provide a more advanced procedure that generates a globally best selection of trial edges, given the identification of the partial stem $S(0)$, with tip $t = u(0)$ and tail $v(0)$. In particular, we seek edges $(u(h), v(h))$ for each $C(h)$, $h \neq 0$, that yield a trial solution that is optimal over the alternatives available (henceforth called an *optimal trial solution*).

The method makes a single pass of the nodes of the SMC, and of edges that meet each node, hence involving $O(n^2)$ effort in a dense graph. For star(i) defined relative to the "best k" edges meeting node i, the method reduces to $O(kn)$ effort, hence to $O(n)$ effort for k constant.

The method has two types of steps, "across cycle" steps and "within cycle" steps. For each node i, we maintain an "across cycle" cost and predecessor node, denoted a_cost(i) and a_pred(i), and also maintain a "within cycle" cost and predecessor node, denoted w_cost(i) and w_pred(i). In contrast to the Local Improvement Method for selecting trial edges, we do not allow S to be degenerate, and hence, assume $S(0)$ contains at least one node. To handle the case of a degenerate S, we extract r from C to compose $S(0)$, with $u(0) = v(0) = r$, and reconstitute C by joining the two previous subroots of r.

Define $N(h)$ to be the set of nodes in $C(h)$, for $h \in H^*$ and $h \neq 0$, and define $N(0)$ to be the set consisting of the single node $v(0)$. The method starts with $h = 0$, and then examines elements of H^* in reverse (predecessor) order, from 0 to -1 to the last element of H^*, and finally ending with the element whose predecessor is the first element of H, if H is non-empty.

SMC Cycle Linking Procedure.

Initialization step. Set a_cost(i) and w_cost(i) to infinity for all nodes $i \neq v(0)$ and set w_cost($v(0)$) = 0. Begin with $h = 0$.

Across cycle step. Let h' denote the predecessor of h in H^*. For each node i of $N(h)$, examine each edge (i, j) such that $j \in \text{star}(i) \cap N(h')$. If $c(i, j) + \text{w\_cost}(i) < \text{a\_cost}(j)$, then set $\text{a\_cost}(j) = c(i, j) + \text{w\_cost}(i)$, and set $i = \text{a\_pred}(j)$.

Within cycle step. Let $h = h'$ (the predecessor element of h in H^*). For each $i \in N(h)$, examine the two edges (i, j) in $C(h)$ meeting node i. If $\text{a\_cost}(j) - c(i, j) < \text{w\_cost}(j)$, then set $\text{w\_cost}(j) = \text{a\_cost}(j) - c(i, j)$ and set $i = \text{w\_pred}(j)$. Finally, if the predecessor of h is 0, go to the Final Step. Otherwise, return to the across cycle step.

Final step. Let $i^* = \text{argmin}(\text{w\_cost}(i, u(0)) + c(i, u(0))\colon\ i \in N(h))$, and let a_cost $(u(0)) = \text{w\_cost}(i^*, u(0) + c(i^*, u(0)))$. The trial edges $(u(h), v(h))$ for each $C(h)$, $h \in H$ and $h = -1$, are identified as follows. Begin with h at its current value (with predecessor 0).

(1) Let $v(h) = i^*$ and $u(h) = \text{w\_pred}(i^*)$.

(2) If $h = -1$, stop. Otherwise, redefine $i^* = \text{a\_pred}(u(h))$, replace h by its successor in H^*, and return to (1).

The following theorem establishes the optimality of this procedure.

Theorem 3. *For the given sequencing of the cycles $C(h)$, $h \in H$, the SMC Cycle Linking Procedure yields an optimal trial solution based on determining $u(h)$ and $v(h)$ for each $h \in H^* - \{0\}$, with $u(0)$ and $v(0)$ fixed. The cost of this trial solution equals:*

$$\text{a\_cost}(u(0) + \text{cost}(S(0)) + \textstyle\sum(\text{cost}(C(h)): h \in H^* - \{0\}),$$

where $\text{cost}(X)$ *denotes the sum of edge costs in subgraph X.*

Proof. The validity of the theorem follows by identifying the linking problem as equivalent to solving an acyclic shortest path problem whose digraph results by creating two layers of nodes, each duplicating the nodes of $N(h)$, for every $h \in H^*$. A collection of *across cycle* arcs (i, j) is created, for each i in the second layer of $N(h)$ and each j in the first layer of $N(h')$, where h' precedes h, over all $h \in H^*$. The cost of each such arc equals the cost $c(i, j)$ of the associated edge of the original problem. The digraph also contains *within cycle* arcs (i, j), for each i in the first layer of $N(h)$ and for each of the two nodes j in the second layer of $N(h)$ such that (i, j) is an edge of $C(h)$, over all $h \in H^* - \{0\}$. Each within cycle arc has a cost of $-c(i, j)$ (associated with dropping edge $c(i, j)$). Evidently, each way to identify edges $(u(h), v(h))$ over all $h \in H^* - \{0\}$, and each way to create linking edges $(u(h'), v(h))$, where h' precedes h, over all $h \in H^*$, corresponds to a unique path from $v(0)$ to $u(0)$ in the acyclic digraph. The SMC Linking Procedure then can be interpreted as a specialization of a standard method for solving an acyclic shortest path problem. □

8. Implications and conclusions

The preceding sections give characterizations and connectivity results for TSP neighborhoods created by ejection chain constructions, which range from generalizations of alternating paths to more complex transformations. In each instance, the key to generating the desired forms of these constructions lies in establishing an associated reference structure as a guidance mechanism. Our designs allow transformations to be carried out in a different space than the space of tours, yet allow access to tours by associated trial solution mappings.

The orientation of this paper has been primarily structural, seeking to identify transformations based on a graph theory perspective, and to disclose their underlying properties. Special algorithmic consequences of these results are explored in a sequel [10] by showing the ejection chain constructions identified here can be exploited by a fast updating procedure that requires only O(1) effort per iteration. In addition, our results give a basis for obtaining a *combinatorial leverage effect*, where the investment of $O(n^2)$ or $O(n^3)$ effort produces solutions that dominate $O(n2^n)$ or $O((n/2)!)$ alternatives, respectively. We also show how this work yields strategies to create combinatorial leverage for NP hard graph problems other than the traveling salesman problem.

In conclusion, we note that recent studies are confirming the usefulness of these ejection chain approaches. Pesch [22] develops a TSP method based on the stem-and-cycle reference structure that performs more effectively than recent extensions of the Lin–Kernighan approach (which combine the L–K method with genetic algorithms). Rego [23] provides an effective application of ejection chain procedures to vehicle routing which likewise proves competitive with currently leading approaches. Finally, by extension of the results of Section 7, Glover and Punnen [11] develop new methods that find optimal tours in linear time over subsets that contain exponential numbers of tours. These outcomes suggest a variety of possibilities exist for further applications of these ideas.

References

[1] C. Berge, Theory of Graphs and its Applications (Methuen, London, 1962).
[2] D. de Werra and F.S. Roberts, On the use of augmenting chains in chain packings, Discrete Appl. Math. 30 (1991) 137–149.
[3] U. Dorndorf and E. Pesch, Fast clustering algorithms, INFORM and University of Limburg (1992).
[4] J. Edmonds, Maximum matching and a polyhedron with 0, 1-vertices, J. Res. Nat. Bureau Standards 69B (1965) 125–130.
[5] J. Edmonds, Matroid intersection, Ann. Discrete Math. 4 (1979) 39–40.
[6] C.-N. Fiechter, A parallel tabu search algorithm for large scale traveling salesman problems, Working Paper 90/1, Département de Mathématiques, École Polytechnique Fédérale de Lausanne, Lausanne (1990).
[7] F. Glover, A bound escalation method for the solution of integer linear programs, Cahiers Centre Études Rech. Opér. 6(3) (1964) 131–168.
[8] F. Glover, Multilevel tabu search and embedded search neighborhoods for the traveling salesman problem, School of Business, University of Colorado, Boulder, CO (1991).
[9] F. Glover, New ejection chain and alternating path methods for traveling salesman problems, in: Balci, Sharda and Zenios, eds., Computer Science in Operations Research: New Developments in Their Interfaces (Pergamon, Oxford, 1992) 491–507.
[10] F. Glover, Ejection chains and combinatorial leverage for traveling salesman problems, School of Business, University of Colorado, Boulder, CO (1992).
[11] F. Glover and A. Punnen, The traveling salesman problem: Linear time heuristics with exponential combinatorial leverage, School of Business, University of Colorado, Boulder, CO (1994).
[12] F. Glover, E. Taillard and D. de Werra, A user's guide to tabu search, Ann. Oper. Res. 41 (1993) 12–37.
[13] B.L. Golden and W. R. Stewart, Empirical analysis of heuristics, in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan and D.B. Schmoys, eds., The Traveling Salesman Problem (Wiley, New York, 1985) 207–250.
[14] F. Harary, Graph Theory (Addison-Wesley, Reading, MA, 1969).
[15] R. Hubscher and F. Glover, Ejection chain methods and tabu search for clustering, School of Business, University of Colorado, Boulder, CO (1992).
[16] M. Laguna, J. Kelly, J.L. Gonzales-Velarde and F. Glover, Tabu search for the multilevel generalized assignment problem, European J. Oper. Res. 82 (1995) 176–189.

[17] E.L. Lawler, Combinatorial Optimization: Networks and Matroids (Holt, Rinehart and Winston, New York, 1976).

[18] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Schmoys, The Traveling Salesman Problem (Wiley, New York, 1985).

[19] S. Lin and B.W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Oper. Res. 21 (1973) 498–516.

[20] M. Malek, M. Guruswany, M. Pandya and H. Owens, Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem, in: F. Glover and H.J. Greenberg, eds., Linkages With Artificial Intelligence, Ann. Oper. Res. 21 (1989) 59–84.

[21] E. Pesch, Personal communications, 1991.

[22] E. Pesch, Reference structures, neighborhoods and candidate lists for TSP ejection chains, University of Maastricht, Maastricht (1993).

[23] C. Rego, Une heuristic tabou avec structures de voisinage composé pour les tournées de vehicules, INRIA, Le Chesnay, France (1993).