Node Ejection Chains for the Vehicle Routing Problem: Sequential and Parallel Algorithms

César Rego [1]

Hearin Center for Enterprise Science, School of Business Administration, University of Mississippi, University, MS 38677, USA

Abstract

We present a Tabu search algorithm for the vehicle routing problem under capacity and distance restrictions. The neighborhood search is based on compound moves generated by a node-ejection chain process. During the course of the algorithm, two types of neighborhood structures are used and crossing infeasible solutions is allowed. Then, a parallel version of the algorithm which exploits the moves' characteristics is described. Parallel processing is used to explore the solution space more extensively and to accelerate the search process. Tests are carried out on a SUNSparc workstation and the parallel algorithm uses a network of four of these machines. Numerical tests indicate that the sequential version of the algorithm is highly competitive with the best existing heuristics and that the parallel algorithm outperforms all of these algorithms.

Key words: Tabu search, vehicle routing, ejection chains, parallel processing.

1 Introduction

The Vehicle Routing Problem (VRP) is a generic name given to a whole class of problems in which a set of routes for a fleet of vehicles based at one or several depots must be determined for a number of geographically dispersed *cities* or *customers*, subject to side constraints. The VRP arises naturally as a central problem in the fields of transportation, distribution and logistics (see Laporte and Osman [29] for a bibliography on vehicle routing problems).

Email address: crego@bus.olemiss.edu (César Rego).

[1] This research was supported in part by ONR grants N000140010598 and N000140010769

The classical VRP is defined as follows. Let $G = (V, A)$ be a graph where $V = \{v_0, v_1, \cdots, v_n\}$ is a vertex set, and $A = \{(v_i, v_j) \mid v_i, v_j \in V; i \neq j\}$ is an arc set. Consider a *depot* to be located at v_0 and let $V' = V \backslash \{v_0\}$ be used as the set of n cities. A matrix C of non-negative *costs* or *distances* c_{ij} is defined on A. When $c_{ij} = c_{ji}$ for all $(v_i, v_j) \in A$ the problem is said to be symmetric and it is then common to replace A with the edge set $E = \{(v_i, v_j) \mid v_i, v_j \in V; i < j\}$.

We assume that m identical vehicles each with capacity Q are used and their number is a decision variable. Vehicles make *collections* or *deliveries* but not both. With each vertex v_i in V' is associated a quantity q_i of some goods to be delivered by a vehicle. The VRP thus consists of determining a set of m vehicle routes of minimal total cost, starting and ending at a depot, such that every vertex in V' is visited exactly once by one vehicle and the total quantity assigned to each route does not exceed the capacity of the vehicle which services the route.

We will also consider an extension of this problem in which a service time δ_i is required by a vehicle to unload the quantity q_i at v_i. It is required that the total duration of any vehicle route (travel plus service times) may not surpass a given bound D, so, in this context the cost c_{ij} is taken to be the travel times between the cities.

The VRP defined above is NP-hard and the aim of this paper is to describe a new *tabu search* algorithm for the general VRP defined above. *Tabu search* is a *metaheuristic* proposed by Glover [13]. The method is generically presented in Glover [14,15] and recent developments may be found in Glover [18,19]. For a comprehensive description of the method and applications see Glover and Laguna [22].

A number of algorithms based on this approach have already been applied to the VRP, each one using different types of moves leading from one solution to another (see, Osman [32], Taillard [39], Gendreau, Hertz and Laporte [11], Rochat and Taillard [37], Rego [35], Xu and Kelly [40]).

An important contribution of our method is the use of embedded neighborhood structures based on the idea of ejection chains. Embedded neighborhoods may be conceived as the outcome of compressing a sequence of moves into a single compound move, and ejection chain procedures give a useful way to build these neighborhoods. For a detailed explanation of ejection chain methods we refer to Glover [20,16,17] and Rego [33]. A number of methods based on this prespective have recently been proposed for various combinatorial problems (see Laguna *et al.* [28], Dorndorf and Pesch [8], Hubscher and Glover [27], Rego [34,35], Glover, Pesch and Osman [23], Cao and Glover [2]).

The remainder of this paper is organized as follows. In section 2, we briefly summarize the ideas underlying ejection chains and we describe their appli-

cation to the VRP. Section 3 describes the sequential version of the proposed algorithm and a parallel approach is described in section 4. Then, the computational results and a comparative analysis of the algorithms are presented in section 5. Finally, section 6 contains a summary and concluding remarks.

2 New neighborhood structures for vehicle routes

A fundamental aspect in the performance of all tabu search application is to identify an effective neighborhood for defining *moves* from one solution to another. Recent tabu search applications to the VRP attempt to increase their performance using compound moves. The basic moves consist of a simple insertion or exchange of vertices/arcs on the graph of the problem and compound moves are usually obtained by combinations of these moves. Osman [32] uses a combination of insertion moves and exchange moves, vertex shifts from one route to another and exchanges vertices between routes based on 2-opt process. Another type of compound move has been used by Gendreau *et al.* [11]. Here, a compound move consists of a simple insertion followed by only one 3-opt exchange or 4-opt exchange. Rochat and Taillard [37] combine partial route constructions with node insertions to obtain complete neighboring solutions. Finally, Rego [35] considers a subpath ejection chain method based on the identification of a reference structure to generate compound moves. In this paper we will use another type of ejection chain process based on the ejection of nodes.

2.1 Node ejection chains for the VRP

In this section, we describe how a solution can be modified to generate another neighboring solution using a node-ejection chain method which can be equally applied to symmetric and asymmetric problems.

Ejection chain procedures have been defined by Glover [20,17] for the TSP and provide an interesting way to generate neighborhoods of compound moves. Here we give an illustration of how these procedures can also be used for the VRP.

We assume that a partial graph $S = (V, X)$ associated with a solution is given. To maintain Glover's perspective, a node ejection chain may be viewed as a series of levels, each consisting of three vertices which appear consecutively in a route.

We will use the following notation. For any vertex v, in a given orientation

of a route, let $\underline{v}$ be its predecessor and $\bar{v}$ be its successor. Consequently, we denote by $(\underline{v}, v, \bar{v})$ a *triplet* representing two consecutive arcs $(\underline{v}, v)$ and $(v, \bar{v})$.

An ejection chain is defined on a subgraph $L = (W, T)$ of S where $T = \{(\underline{v}^0, v^0, \bar{v}^0), \cdots, (\underline{v}^k, v^k, \bar{v}^k), \cdots, (\underline{v}^l, v^l, \bar{v}^l)\}$ is a set of triplets representing $l+1$ levels of an ejection chain, which we denote by

$$T = \bigcup_{k=0}^{l} \Big\{(\underline{v}^k, v^k, \underline{v}^k)\Big\}.$$

An ejection chain can be completely determined by a succession of *central vertices*, $v^0, \cdots, v^k, \cdots, v^l$, which we designate by the set Γ^l; v^0 is the *top vertex* and v^l is the *bottom vertex*. Consequently, we denote by $\underline{\Gamma}^l$ and $\overline{\Gamma}^l$ respectively the set of predecessors and successors of vertices in Γ^l. Also we denote by W^l the set of vertices in all triplets of the ejection chain (i.e. $W^l = \underline{\Gamma}^l \cup \Gamma^l \cup \overline{\Gamma}^l$).

An *ejection* results by moving a vertex to a new position occupied by another vertex, disconnecting this vertex from its position.

Let k be a level of the chain, in an ejection chain each vertex v^k ejects the vertex v^{k+1} ending with the ejection of the v^l.

As result, an ejection chain process of $l+1$ levels is a replacement of T by

$$T' = \bigcup_{k=1}^{l} \Big\{(\underline{v}^k, v^{k-1}, \bar{v}^k)\Big\}.$$

transforming S into a disconnected graph. In other words, triplets $(\underline{v}^k, v^k, \bar{v}^k)$ $(k = 0, 1, \cdots, l)$ are successively replaced by triplets $(\underline{v}^k, v^{k-1}, \bar{v}^k)$, $(k = 1, 2, \cdots, l)$. Because v^l vertex is not attached to any route, this transformation does not represent a complete transition from the current route set to a new route set.

The complete transition may be obtained by two *connectivity processes*:

(a) **Type I (connectivity process)** which consists of creating the set
$$T'_I = \Big\{(\underline{v}^0, v^l, \bar{v}^0)\Big\};$$

(b) **Type II (connectivity process)** which consists of choosing a set
$$\bar{T}' = \{(v^l_p, v^l_q)\}$$
in $\overline{L} = (\overline{W} = V \setminus \Gamma^l, \overline{T} = X \setminus T)$, and replacing it by another set
$$T'_{II} = \Big\{(\underline{v}^0, \bar{v}^0), (v^l_p, v^l, v^l_q)\Big\}.$$

That means, we get a new neighboring solution $S'_I = S \cup T' \cup T'_I \setminus T$ in the first case and $S'_{II} = S \cup T' \cup T'_{II} \setminus (T \cup \overline{T'})$ in the second case. Illustrative diagrams of these concepts are given in Figure 1 for three levels of an ejection

chain. Dotted lines represent the set T in the left-hand diagram and the sets T and $\overline{T'}$ in the right-hand diagram. Similarly, black lines designate the sets $T^{'}$ and $T_I^{'}$ in the left-hand diagram and the sets $T^{'}$ and $T_{II}^{'}$ in the right-hand diagram.

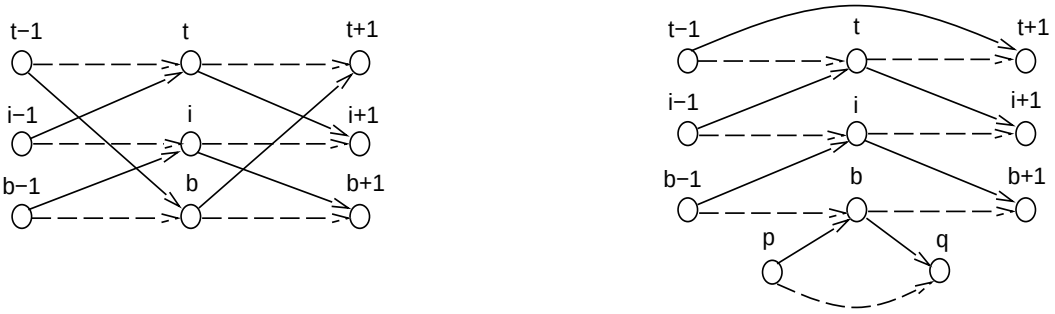

Fig. 1. Two types of neighborhood structures

These characterizations of neighborhood structures make it possible to define compound moves to generate neighboring solutions. These moves can be divided into two types of processes:

(i) **multi-node exchange process** which is represented by an ejection chain ended with the Type I connectivity process. Figure 2 shows an example of 5 levels of an ejection chain in which $T = \{(0, 1, 13), (0, 5, 6), (6, 4, 3), (3, 2, 0), (0, 14, 0)\}$, $T^{'} = \{(0, 1, 6), (6, 5, 3), (3, 4, 0), (0, 2, 0)\}$, and $T_I^{'} = \{(0, 14, 13)\}$. Note that this move maintains a constant number of vertices in each route.

(ii) **multi-node insert process** which is represented by an ejection chain ended with the Type II connectivity process. Figure 3 shows an example of 4 levels of an ejection chain in which $T = \{(0, 14, 0), (0, 1, 13), (0, 5, 6), (6, 4, 3)\}$, $T^{'} = \{(0, 14, 13), (0, 1, 6), (6, 5, 3)\}$, $\overline{T'} = \{(3, 2)\}$ and $T_{II}^{'} = \{(0, 0), (3, 4, 2)\}$. Note that in this move an interesting phenomenon occurs when a vertex 14 is chosen to be the top vertex of the chain. This means that the number of vehicle routes will be reduced. Also, the inverse may occur (i.e., a new vehicle route can be created) if the bottom vertex is inserted into a degenerated route. In this case v_p and v_q are the same vertex, the depot.

2.2 Evaluating the cost of a move

In order to evaluate a compound move efficiently, we require that the original costs (associated with the ejection values) are not modified during the ejection chain construction. To do that, it is still necessary to define a condition to prevent an arc from being inserted more than once.

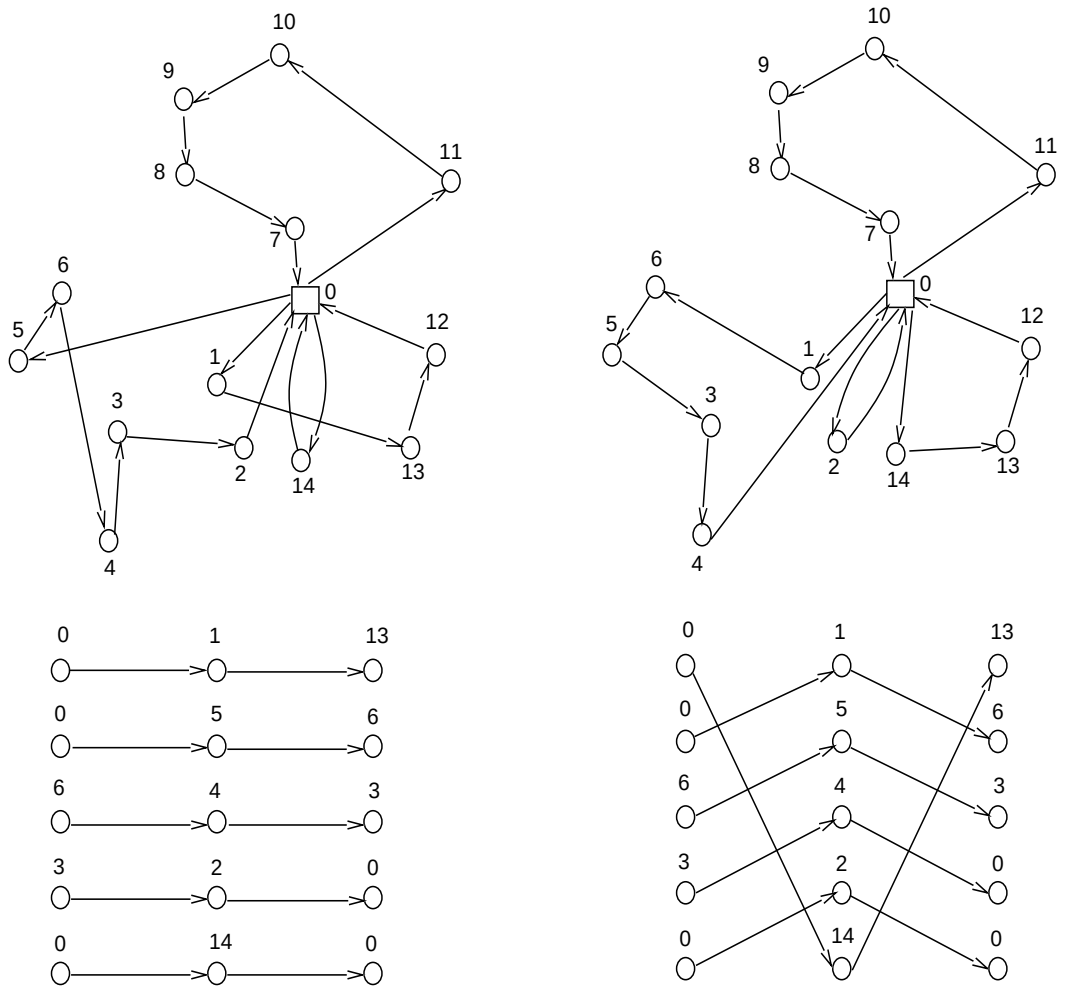

Fig. 2. Multi-node exchange process

This condition is imposed by a so-called *legitimacy restriction* stipulating that each vertex in Γ^l occurs only once in W^l. Moreover, any predecessor of a vertex in Γ^l may reappear as a successor of a vertex in Γ^l and vice versa, without violating this restriction. However, an exception is made for arcs linked to the depot which can reappear several times but only connected to vertices in Γ^l, that is, vertex v_0 cannot be ejected. We say that an ejection chain satisfying this *legitimacy condition* has a *legitimate structure*.

To evaluate the change in solution cost created by a compound move at a given level k of an ejection chain, two types of *trial moves* associated with each type of connectivity process are used.

Type I (trial move) Inserting the current bottom vertex between the last predecessor and successor of v^0, creating arcs $(\underline{v}^0, v^k)$ and $(v^k, \bar{v}^0)$.

Type II (trial move) Creating the arc $(\underline{v}^0, \bar{v}^0)$ and choosing two vertices $v_p, v_q = \bar{v}_p \notin \Gamma^k$. Then, inserting v^k between v_p and v_q, adding arcs (v_p^k, v^k), (v^k, v_q^k) and deleting the arc (v_p, v_q).

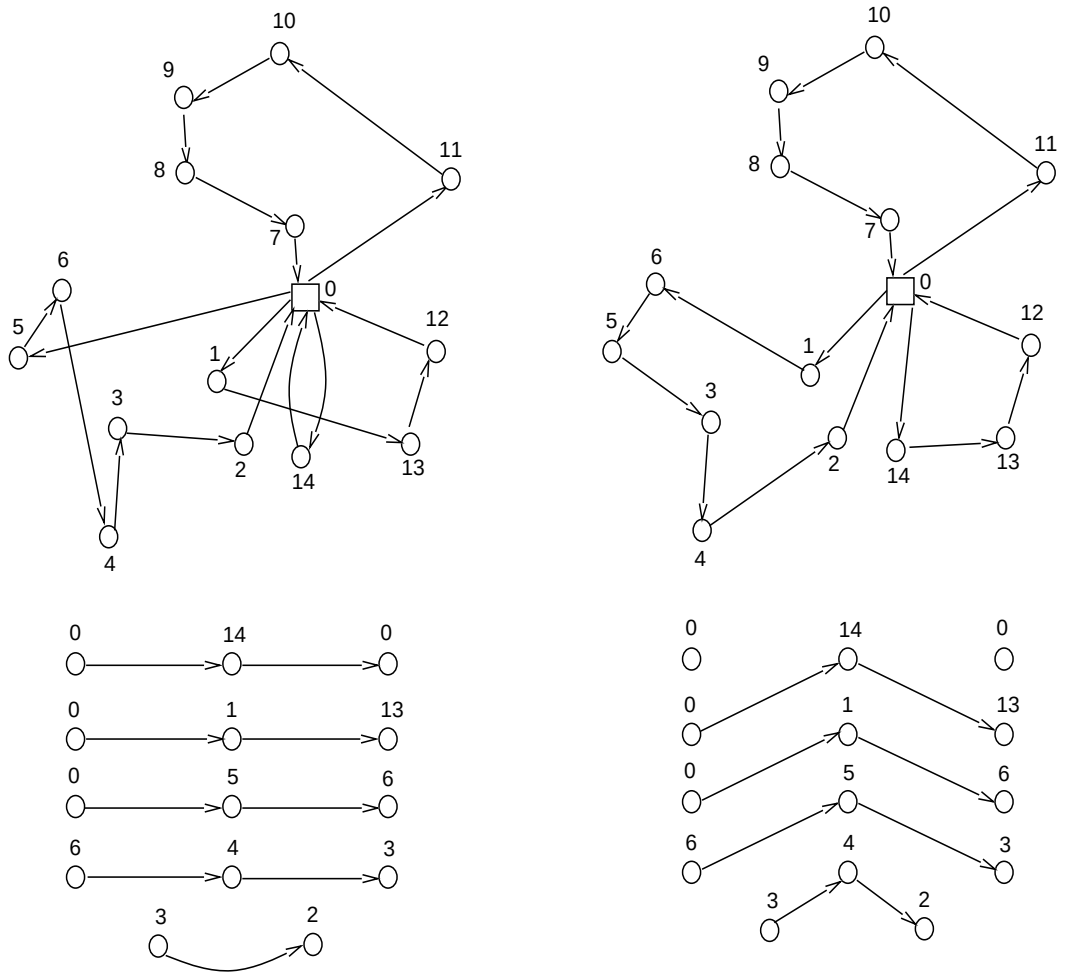

Fig. 3. Multi-node insert process

For convenience, we will denote the $c_{ij}s$ values by $c(v_i, v_j)$ and consequently, let us define the cost of a triplet as the cost sum of its arcs, $c(v_i, v_j, v_k) = c(v_i, v_j) + c(v_j, v_k)$.

Thus, an ejection chain process of l levels may be recursively evaluated as follows:

$$e_k = \begin{cases} \min\limits_{v_i, v_j \in V'} \{c(\underline{v}^k, v^{k-1}, \bar{v}^k) - c(\underline{v}^{k-1}, v^{k-1}, \bar{v}^{k-1}) \\ \qquad\qquad -c(\underline{v}^k, v^k, \bar{v}^k) + \lambda\ c(\underline{v}^{k-1}, \bar{v}^{k-1})\}\ k = 1 \\ \\ e_{k-1} + \min\limits_{v_i, v_j \notin W^{k-1}} \{c(\underline{v}^k, v^{k-1}, \bar{v}^k) - c(\underline{v}^k, v^k, \bar{v}^k)\} \qquad 1 < k \leq l \end{cases}$$

where $\lambda = 1$ if Type II trial move is used and $\lambda = 0$ otherwise.

Denote by *trial value*, the cost change obtained by a trial move, which may be determined as:

$$\Delta^k = \begin{cases} e_k + c(\underline{v}^0, v^k, \bar{v}^0) & \text{if Type I trial move,} \\ \\ e_k + \min\limits_{v_p \in V' \setminus \Gamma^k} \{c(v_p, v^k, \bar{v}_p) - c(v_p, \bar{v}_p)\} & \text{if Type II trial move,} \end{cases}$$

where Δ^k is the change in the *Objective* function value before and after to make k levels of an ejection chain ending with Type I or Type II trial moves. The solution associated with the move is called *trial solution*

3 The tabu search implementation

We present a description of the TABUCHAIN algorithm. We first describe the neighborhood search procedure used in the algorithm. Here, the main feature is the use of embedded neighborhood structures built by an ejection chain strategy.

3.1 The neighborhood search procedure

By definition, a compound move obtained by an ejection chain construction results in successively simple neighborhoods exploited at each level of the chain.

For any vertex v in V', we define its h-neighborhood $N_h(v)$ as the set of the h vertices closest to v, and consequently we define its *legitimate neighborhood* as the subset of these vertices that do not violate the legitimacy restriction. Thus, for a given level k of an ejection chain, we define the *legitimate neighborhood* of a vertex v^k as the set:

$$LN_h(v^k) = \{v_j \mid v \in N_h(v^k), v \notin W^{k-1}\}$$

Note that the set $LN_h(v^k)$ becomes more restricted when k increases, because the requirements of legitimacy reduce the number of choices at each level.

The chain is initialized by the choice of the v^0 vertex and the associated initial v^1 vertex.

The chain grows by a bottom extension move. This move selects a vertex $v \in LN(v^1)$ to become the new bottom vertex, by disconnecting v from its current position and relocating v^1 to occupy this position.

The general neighborhood search procedure can now be described as follows.

PROCEDURE *NodeEjectionChain (current solution: S, number of levels: l)*

- Initialize legitimate neighborhood for all vertices and set $k = 0$.
- Determine a set of two vertices $\{v^k, v^{k+1}\} = arg\{e_{k+1}\}$ which denote respectively the top vertex and the initial bottom vertex of the ejection chain.
- Set $\Gamma^k = \{v^k\}$.
- While $LN_h(v^{k+1}) \neq \emptyset$ and $k < l$ do
  - Set $k = k + 1$ and set $\Gamma^k = \Gamma^{k-1} \cup \{v^k\}$.
  - Inspect the cost changes by calculating the corresponding trial value Δ^k.
  - Update the best level k^* so far that yielded the minimum trial value. If using the type II trial move then record the associated v_p^k vertex.
  - Determine a new vertex $v \in LN_h(v^k)$ by computing e_k, and set $v^{k+1} = v$.
  - Update the legitimate neighborhoods for every vertex $v_i \in W^{k^*}$.
- Set $l = k^*$ and update the current solution according to Type I or Type II trial moves.

Complexity

The *complexity* of this procedure is determined as follows. To start an ejection chain, $O(n^2)$ choices of v^0 and v^1 would be considered. For the remaining levels of the chain the new bottom vertex is selected in $O(n)$ time. It is clear that the maximum length of an ejection chain is limited by the legitimacy restriction and depends on the number of times that vertex v_0 (depot) appears in the chain. Let r be the number of routes in a given solution, then the maximum number of levels of a legitimate ejection chain is bounded by $O((n + r)/2)$.

Considering first the Type I trial move, the bottom vertex is relocated in $O(1)$ time. Then it is possible to perform one iteration of the algorithm in $O(n^2 + (n + 1)\frac{n+r}{2})$.

For the evaluation of the Type II trial move we have to choose the best insertion among $n + r$ vertices. In this case, the move may be evaluated in $O(n^2 + (2n + r)\frac{n+r}{2})$.

From a practical standpoint, this effort can be notably reduced because it is not usually necessary to perform all possible levels of the chain to obtain the best move; therefore it may be limited by a parameter which we denote by l_{max}.

The value of r depends on the type of VRP, but naturally may be defined as:

$$r = \left\lceil \frac{\sum_{i \in V \setminus \{v_o\}} q_i}{Q} \right\rceil + \nu$$

where ν is an integer value. If cities are uniformly distributed, ν is a very small number and as generally, for a given type of a problem r is much smaller than n and at most equal to n, then *the overall complexity of both moves is* $O(n^2)$.

3.2 The TABUCHAIN algorithm

The main algorithm can now be described. It starts with an initial solution given by the parallel version of the Clarke and Wright [5] procedure. This procedure starts with vehicle routes containing the depot and one other vertex. Satisfying the demands of cities i and j, the cost of a solution is reduced. The cost saving of visiting cities i and j using one vehicle can be obtained by $\xi_{ij} = c_{i0} + c_{0j} - c_{ij}$ (see Figure 4). At each step, two routes are merged according to the largest saving ξ_{ij} without violating the problem restrictions, until no further merges are possible.

Fig. 4. Cost savings of merging two routes

The idea of using a route construction heuristic has already been used by a number of authors (see, Osman [32], Gendreau *et al.* [11]) and the aim is to rapidly generate an initial feasible solution. We consider the classic parallel version of the savings procedure with a time complexity of $O(n^2 \log n)$.

TABUCHAIN makes use of two procedures, One-Search and Two-Search, which use respectively two types of compound moves defined in section 2.1 to improve successively the pre-existing routes. Initially, the goal of TABUCHAIN is to converge quickly to a good solution as well as creating a knowledge base to be used in further phases of the algorithm. In the framework of tabu search, this is possible using attribute based memory structures. We will now describe how the algorithm creates and uses these memory structures.

Short term memory

Short term memory functions are specified by tabu restrictions and aspiration criteria. To avoid cycling, some *attributes* of a move are stored in a tabu list rather than the complete visited solutions. For the management of the tabu list, it is necessary to define two criteria:

(i) a *forbidden criterion* that defines which attributes to put into the tabu list,
(ii) a *cancellation criterion* which defines when an attribute leaves the tabu list.

Osman [32], Taillard [39] and Gendreau *et al.* [11] have considered attributes (i, r) which indicate that a given vertex v_i was shifted out of the route r. The tabu list is made up of these attributes which forbid a vertex from being inserted in a route. This means that exchange moves are forbidden if at least one attribute is in the tabu list.

In our application, we use attributes (i, j) specifying *transition moves* (ejections) from one level to another within an ejection chain. In theoretical terms, the inverse ejection (j, i) (for all levels up to k^*) as well as the ejections of the ejection chain itself should be forbidden. This is because some combinations of ejections would lead to cycling. However, several tests have shown this type of tabu restriction to be very restrictive. In practice, to avoid cycling it is sufficient to forbid the ejection of the first level of the chain and the inverse ejection of the last level.

Usually, the cancelation criterion consists of freeing attributes from the tabu list at the end of a certain number θ of iterations, where θ is known by the tabu list size or *tabu tenure*. The θ value depends on the problem characteristics and practical experiments indicate that changing this value randomly (see, e.g., Taillard [39]) or systematically (see Chakrapani *et al.* [3]) during the search is often more favorable than having a fixed value. Osman [32] uses a more sophisticated process based on regression analyses that exploits some problem characteristics.

In our application, each transition move receives a tabu tenure value chosen randomly between θ_{min} and θ_{max}.

Nevertheless, if at a given time an ejection move is declared to be tabu, this status can be disregarded if the corresponding trial move improves the best solution found so far, which defines the classical *aspiration function* used in Tabu search:

$$C(S) + \Delta^k < C(S^*).$$

Intermediate and long term memory

Intermediate and long term memory are applied as follows. If a better solution is obtained as a result of two successive iterations and the solution on the next iteration is not an improvement, then a number of complete ejection chains are generated from the best current solution in order to intensify the search locally. (The chains are started with the least cost ejection moves.) This has the advantage of avoiding the loss of information throughout the search.

Since the algorithm chooses the highest evaluation move in each iteration, then the frequency of a move may establish a measure of its attractiveness. More significantly, this measure may be expressed in terms of its most intrinsic attributes, such as the node links modified by the move. Clearly it varies according to the current search state and therefore may be interpreted as a dynamic measure of *influence* (see Glover and Laguna [21]).

For a given set of arcs $A^{'} \subset A$, let us denote $F(A^{'})$ as being the sum of the frequencies of every arc of $A^{'}$. Consequently, in our application the influence of a move m of level k on the quality of the solution depends on the type of move and is given as:

$$I(m_k) = \begin{cases} F(T^{'}) - F(T) & \text{if multi-node exchange move,} \\ \\ F((\underline{v}^0, \bar{v}^0)) + F(T^{'}) - F(T) & \text{if multi-node insert move.} \end{cases}$$

Indeed, we use frequency based memory specified by these influence measures in order to perturb the *Objective* function in different phases of the search. Thus, these influence values introduce *incentives* for incorporating "good attributes" in an intensification phase and *penalties* in a diversification phase.

Incentive and penalty functions are as follows:

$$C(S_k') = C(S) + \Delta^k \pm \ln\left(1 + I(m_k)\right) (+ \text{ diversification}, - \text{ intensification}).$$

Here, the logarithmic function is used to soften the weight of frequencies ensuring that it does not excessively perturb the evaluation of other candidate moves.

The diversification strategy is implemented by periodically activating the penalty function in order to force the search to explore new regions. This

strategy has the advantage of making continuous use of the historical information. Alternatively, another diversification strategy could be used which would stop the search and restart it from an unexplored solution that contains, for example, a number of arcs that have not yet been modified.

Strategic oscillation

In a general context, Glover [14,15] proposes the use of an interplay between intensification and diversification called strategic oscillation. In our application, In our application, we basically use two different forms of strategic oscillation. In the first approach, we apply the version of this strategy that induces the exploration of new regions by crossing certain boundaries between feasibility and infeasibility. This idea is also used by Gendreau, Hertz and Laporte [11] in the VRP context.

We define a *move* to be *infeasible* if it causes a violation of the capacity or maximal length constraint. Our strategic oscillation differs from that proposed by Gendreau et al. because no transition to an infeasible trial solution is accepted. However, feasible trial solutions can be obtained by combining feasible and infeasible ejection moves in the ejection chain construction: this avoids the use of penalty factors to guide the search toward a feasible solution. This procedure is based on the fact that infeasible ejection moves, when properly controlled, can often lead to feasible compound moves at further levels in the chain.

More formally, consider two vehicle routes R_s and R_w such that $v_k \in R_s$ and $v^{k+1} \in R_w$, and let Q_s, Q_w denote respectively the total demand associated with the set of vertices in routes s and w. Suppose now a situation where at level k of the ejection chain an ejection move replaces vertex v^{k+1} by vertex v^k, hence becoming $v^k \in R_w$. As a result, it might happen that after inserting vertex v^k into R_w, the vehicle capacity of route R_w is exceeded, i.e. $Q_w + q_k - q_{k+1} > Q$. However, it is possible that in a next level of the chain another vertex of route R_w is replaced by a "lower demand" vertex from other route, so that the (negative) difference between the quantities associated with these vertices restores the feasibility of this route (relative to the vehicle capacity). We should note that for levels of the ejection where the associated ejection move leads to an infeasible vehicle route, no trial move is evaluated since infeasible trial solutions are not allowed, as already mentioned.

In order to avoid complete chains where all intermediate trial moves are infeasible, only feasible trial solutions are accepted in the first level of the chain (i.e., both ejection and trial moves must be feasible). At the end of the ejection chain the feasible highest evaluated trial solution is chosen to make a move.

Another strategic oscillation approach is related to the use of "switching of

moves" (Rego [33]) provided by type I and type II ejection chain neighborhoods. Here, if a given type of move produces an improved solution, then the other one starts with the best encountered solution. Otherwise, the last modified solution is given to the next move and the search goes on in a continuous fashion. This strategy of changing to an alternative neighborhood structure when a local optimum is found is a form of strategic oscillation approach based on critical event memory. See Glover and Laguna [22] for other and more advanced uses of critical event memory.

Post-optimization procedure

A post-optimization procedure consists of a local reoptimization of every route by solving the corresponding TSP. We require this to be a very quick procedure, and therefore we call One-Search procedure for each individual route.

The post-optimization procedure is called in the following cases:
(a) when a solution provides less than 2% above the best solution found so far;
(b) when an improvement is not found in $n_{max}/2$ iterations.

It is clear that when the post-optimization procedure is called in successive iterations, only different routes that contain vertices v in Γ^{k^*} must be reoptimized.

Figure 5 shows an example where two routes must be reoptimized. In this example, two independent ejection chains are applied to each of the routes. For route 1, $T = \{(0,1,5),(2,3,0),(5,6,4)\}$, $T^{'} = \{(2,1,0),(5,3,4)\}$, $T_I^{'} = \{(0,6,5)\}$ and for route 2, $T = \{(9,8,11),(7,10,9)\}$, $T^{'} = \{(7,8,9)\}$ and $T_I^{'} = \{(9,10,11)\}$.

The algorithm calibration

We now comment on some algorithmic aspects of TABUCHAIN.

With respect to the search process, both One-Search and Two-Search procedures are used in all the phases of the algorithm. However, Two-Search is the most important procedure of TABUCHAIN and carries out more than 70% of the search.

Sensitivity analysis were performed on all the test problems in order to calibrate the algorithm parameters. We did not find the algorithm to be very sensitive to these values with respect to the solution quality. Nevertheless they

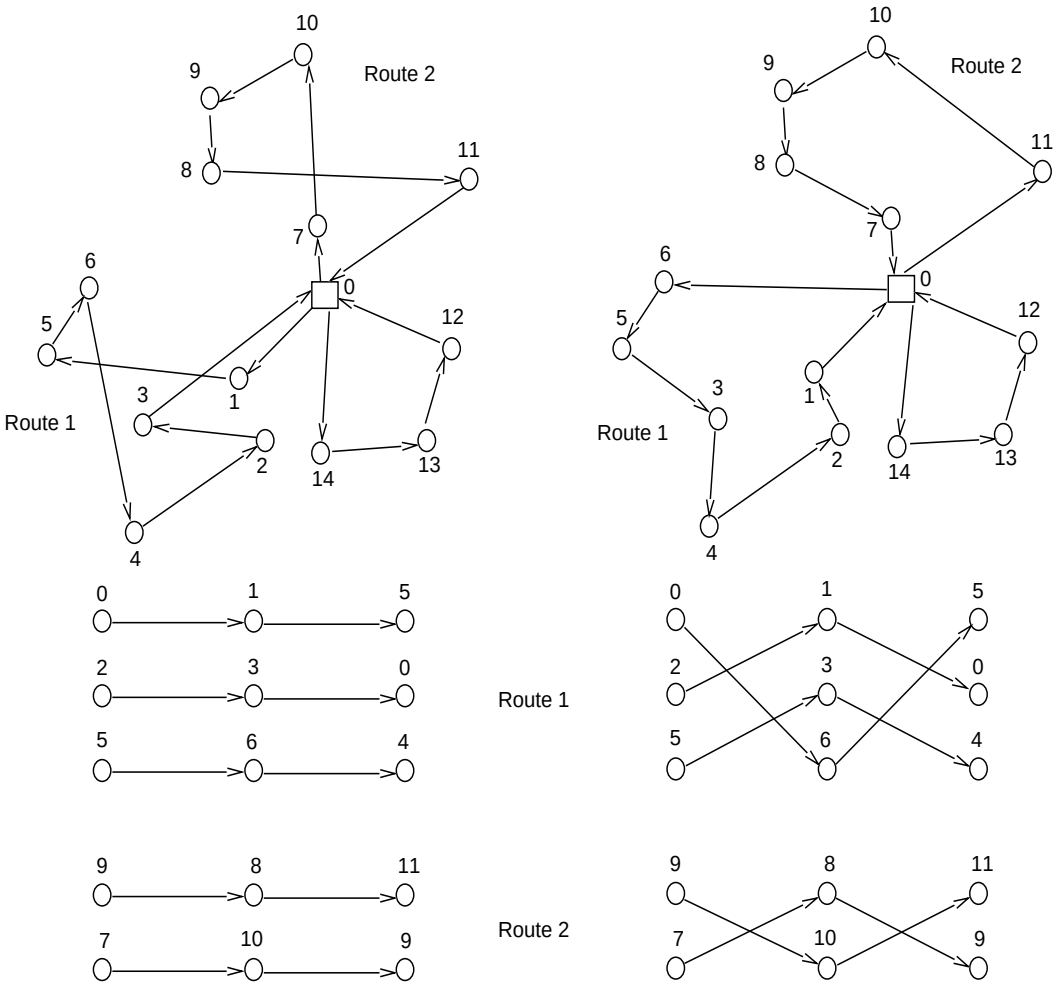

Fig. 5. Multi-node exchange in independent routes

have a significant influence on the computation times, because if the parameters are not adequate for the type of move and the problem characteristics an extended number of iterations would be required to reach equivalent solutions. After extensive experimentation we have verified that the best results are always obtained with parameters which lead to the greatest improvement on the initial solution in $2n$ iterations. When several possible solutions have less than 1% difference between them, priority is given to the one whose configuration differs most from that of the initial solution, which can be empirically evaluated by the total number of ejection chain levels performed so far. This fact may be due to the significant difference between the clusters given by the Clarke and Wright algorithm and those in the optimal solution.

Regarding the tabu list size, the most appropriate values depend on the search strategy and have been obtained as follows. For a normal and intensification phase the best θ_{min} and θ_{max} bounds are found within the interval $[5, 20]$, but when the oscillation and diversification phases are activated, these values are contained in the interval $[80, 120]$. In contrast, for a post-optimization phase

$\theta_{min} = 3$ and $\theta_{max} = 7$ are always used throughout the search.

Also, we have fixed $l_{max} = 6$, except in oscillation and diversification phases, where this value is gradually increased up to the level limited by the legitimacy restrictions. Finally $n_{max} = 20n$ is often sufficient to reach our best solutions but for some instances this value may go up to $50n$, where n is the problem size.

4 Parallel tabu search implementation

The algorithm considers a node-ejection chain process described in the previous section. In this section we show that such ejection chains in conjunction with parallel processing gives a powerful Tabu Search algorithm for the VRP, which outperforms the sequential approach. In the algorithm, parallel processing is used to explore the solution space more extensively, as well as to accelerate the move evaluation in the ejection chain construction.

4.1 The Parallel TABUCHAIN algorithm

Our algorithm was implemented on a network of SUNSparc workstations using the PVM (Parallel Virtual Machine) system, which permits this network to be viewed as a single parallel computer. The unit of parallelism in PVM is a *task*, and (as for a Unix process) multiple tasks may execute on a single processor. A standard model of message passing is used to allow communication and synchronization between tasks.

We use a classic *master-slave* model without communication among the slave processes. Each slave executes a complete tabu search algorithm with a different set of parameters, but from the same starting solution which is given by the Clarke and Wright [5] algorithm. Basically, the work of the master is to collect the best local solution from each slave process and transmit the best among them to slaves, to start up a new iteration of the method. Furthermore, the communication and synchronization processes are controlled by the master process. In accordance with the taxonomy of parallel Tabu search algorithms proposed by Crainic, Toulouse and Gendreau [6] our algorithm can be classified as a 1-control, knowledge synchronization, SPDS (*Single Point Different Strategies*) method with one main process and four child processes. In order to minimize the communication times, only solution values are transmitted to the master and only slaves that did not find the best value receive the corresponding solution.

In our PVM implementation, the algorithm makes use of the following messages for handling the search and the communication process.

- MAKE_SEARCH to a slave process to perform a search.
- END_SEARCH to a slave process to stop the search.
- ELECTED to indicate that a slave process has found the best global solution.
- NOT_ELECTED to indicate that a slave process has not found the best global solution.

The MASTER and SLAVE algorithms are described respectively in as follows.

MASTER algorithm

- Start up slaves tasks.
- Read problem instance and broadcast it to slave tasks.
- Read starting solution.
- Initialize the set of *not elected slaves* as all slave tasks.
- While *an improvement is found* and *the number of restarts are not met* do
  - Broadcast MAKE_SEARCH message to all slave tasks.
  - Send the current best solution to *not elected slaves*.
  - Wait for best local solution values from all slave tasks.
  - Identify the new best global solution.
  - If it is a global improvement then
    Update the best global solution from one of the slaves that has found it, and send an ELECTED message to it asking for the corresponding solution.
    Identify the new set of *not elected slaves* and send a NOT_ELECTED message to them.
    Receive the new best global solution.
  - Otherwise broadcast a NOT_ELECTED message to all slave tasks.
  - Update the number of restarts
- Broadcast an END_SEARCH message to all slave processes
- Record the best global solution found by the algorithm and terminate the search.

SLAVE algorithm.

- Receive message from MASTER.
- While message is equal to MAKE_SEARCH do
  - Identify parameters according to its own slave *task identification* and perform a tabu search algorithm using the *NodeEjectionChain* procedure.
  - Send best solution value to MASTER
  - Receive message to know if its solution was selected
  - If message is equal to NOT_ELECTED
    Receive message from MASTER for continuing or terminating the search

If message is equal to MAKE_SEARCH, receive best global solution from MASTER

· If message is equal to ELECTED

Send best local solution to MASTER

Receive message for continuing or terminating the search

The evaluation of a trial move in a given level of the ejection chain depends on the vertices which were ejected up to the level in question. However, the choice of the ejection move is independent of the trial move evaluation throughout the ejection chain. This property allows these two operations to be performed in parallel, which is important when the trial move evaluation requires a considerable effort. This is the case for the Type II trial move in which at each level of the chain the best insertion of the ejected vertex would be chosen after $O(n)$ comparisons, which is also the effort necessary to determine each ejection move. Thus, when the Type II trial move is active in the *NodeEjectionChain* procedure, the evaluation of each trial solution is kept for another process.

In addition, the algorithm includes a post-optimization procedure, which consists of a local reoptimization of every route by solving the corresponding traveling salesman problem. In the post-optimization procedure, each individual route is assigned to a different process and reoptimized separately by using ejection chains with Type I trial moves.

5 Computational experience

5.1 Characteristics of test problems

The performance of our algorithms was tested on a set of fourteen benchmark problems described in Christofides, Mingozzi and Toth [4].

Problem sizes range between 50 and 199 cities in addition to the depot. Locations of cities are defined by coordinates and the travel cost c_{ij} from city i to city j is a Euclidean distance. Problems may include capacity and maximum route time constraints. For almost all these problems the cities are regularly distributed around the depot, but for problems C11 to C14, they appear in clusters and the depot is not centered.

5.2 Computational results and comparative analysis

Our computations were performed on a 33MHz Sun IPC workstation, and the parallel algorithm runs on a network of four of these machines. Both

codes were written in C and the parallelism was supported by using PVM library routines. Our solution values were calculated with real distances and computational results for the sequential (S-TC) and parallel (P-TC) versions of the TABUCHAIN algorithm are reported in Table 1.

Table 1
Characteristics of test problems, solution values and relative percentage deviation (RPD) for both sequential (S-TC) and parallel (P-TC) versions of the algorithm.

					Best	Solution values		RPD	
Prob.	n	Q	D	δ	published	S-TC	P-TC	S-TC	P-TC
C1	50	160	∞	0	***524.61** [24]	**524.61**	**524.61**	0.00	0.00
C2	75	140	∞	0	**835.26** [39]	837.50	835.32	0.27	0.01
C3	100	200	∞	0	**826.14** [39]	827.53	827.53	0.17	0.17
C4	150	200	∞	0	**1028.42** [39]	1054.29	1044.35	2.52	1.55
C5	199	200	∞	0	**1291.45** [37]	1338.49	1334.55	3.64	3.34
C6	50	160	200	10	**555.43** [39]	**555.43**	**555.43**	0.00	0.00
C7	75	140	160	10	**909.68** [39]	**909.68**	**909.68**	0.00	0.00
C8	100	200	230	10	**865.94** [39]	868.29	866.75	0.27	0.09
C9	150	200	200	10	**1162.55** [39]	1178.84	1164.12	1.40	0.14
C10	199	200	200	10	**1395.85** [37]	1420.84	1420.84	1.79	1.79
C11	120	200	∞	0	**1042.11** [39]	1043.54	**1042.11**	0.14	0.00
C12	100	200	∞	0	***819.56** [9]	**819.56**	**819.56**	0.00	0.00
C13	120	200	720	50	**1541.14** [39]	1550.17	1550.17	0.59	0.59
C14	100	200	1040	90	**866.37** [39]	866.53	**866.37**	0.02	0.00
Average								0.77	0.55

Bold characters correspond to the best known solutions and asterisks indicate values which had already been proved to be optimal. The numbers inside brackets represent the references in which the solutions were obtained.

Analyzing the Sequential TABUCHAIN algorithm

We can see that identical values of solutions were found for 4 problems and that generally the S-TC algorithm produces high quality solutions in a relatively short computation time.

Note that a great deal of computation time has been spent by powerful algo-

rithms in order to find very high quality solutions for these problems.

Analysing the Parallel TABUCHAIN algorithm

We can see that the parallel version never finds solutions worse than the sequential one, and improves **seven** of these solutions. As a result the P-TC algorithm gives solutions which are on average 0.22% better than the S-TC algorithm, which is very significant when we consider that the best known solutions are already optimal or very close to this value.

Comparisons with alternative algorithms

Comparisons were also made between our algorithms and other heuristic algorithms from the literature. It is shown in Gendreau *et al.* [11] that "classical" methods to the VRP, *constructive algorithms* (Clarke and Wright [5], Mole and Jamesson [30], Altinkemer and Gavish [1], Desrochers and Verhoog [7]), *two-phase algorithms*, (Gillett and Miller [12], Christofides, Mingozzi and Toth [4], Fisher and Jaikumar [10]), *incomplete optimization algorithms* (Christofides, Mingozzi and Toth [4]), and *descent improvement algorithms*, (Stewart and Golden [38], Harche and Raghavan [26]), are not on a par with simulated annealing and tabu search algorithms. In addition to the above methods, we may include the recently improved version of the Gillett and Miller [12] algorithm by Renaud, Boctor and Laporte [36]. None of these algorithms has produced solutions less than 2% on average above the best known solutions and are generally not robust. However, some of these algorithms are able to produce different solutions very quickly based on different parameters. Thus, they are useful either for interactive routing systems or to provide initial solutions for improvement methods.

Here, our objective is to compare algorithms of very high performance in terms of solution quality and computation time.

Thus, we have only made comparisons between our algorithms and the best tabu search algorithms for the VRP, whose computation times for finding their best solutions have been published. Computational results for these algorithms are reported in Table 2.

We should note that in the literature, some other important tabu search algorithm exists for the VRP. These are for example Taillard's algorithm [39] and its probabilistic variant proposed in Rochat and Taillard [37], which have found all the best known solutions. However, these solution values are obtained on an unspecified number of runs, and the corresponding computation times

Table 2
Computation times (in seconds) and relative percentage deviation (RPD).

Algorithm	OTS		GHL		S-TC		P-TC	
Computer	VAX 8600		Silicon 4D/35		SUNSparc 4		4 SUNSparc 4	
Problem	CPU	RPD	CPU	RPD	CPU	RPD	CPU	RPD
C1	61	0.00	84	0.00	51	0.00	63	0.00
C2	50	1.05	2352	0.06	1008	0.27	2603	0.01
C3	895	1.07	408	0.40	2034	0.17	1579	0.17
C4	1761	1.55	3270	0.75	1632	2.52	2908	1.55
C5	1704	3.34	5028	2.42	975	3.64	4624	3.34
C6	63	0.00	468	0.00	190	0.00	143	0.00
C7	745	0.15	1908	0.39	1386	0.00	1234	0.00
C8	1965	0.09	354	0.00	516	0.27	1136	0.09
C9	2475	1.85	1278	1.31	933	1.40	1791	0.14
C10	4025	1.58	2646	1.62	3121	1.79	2563	1.79
C11	780	0.00	714	3.01	378	0.14	674	0.00
C12	340	0.00	102	0.00	73	0.00	94	0.00
C13	1576	0.38	2088	2.12	120	0.59	117	0.59
C14	582	0.00	1782	0.00	565	0.02	1479	0.00
Average	1216	0.79	1606	0.86	927	0.77	1501	0.55

OTS: Osman's [32] tabu search algorithm.
GHL: the Gendreau, Hertz and Laporte [11] tabu search algorithm.

are not reported by the authors. Also, the GHL column reports results for the version of the algorithm for which computation times have been published.

For visualization purposes, diagram A of Figure 6 illustrates the solution quality of the P-TC algorithm compared with those reached by the S-TC algorithm. Similarly, diagram B in the same figure gives an impression of the time consumed by these algorithms to find "identical" solution values as well as the time required for the P-TC algorithm to improve these solutions when they occur.

We can see that for some instances the P-TC algorithm takes more time than the S-TC one, but this is compensated for by the quality of solutions.

Also, although different machines were used it is clear that the S-TC algorithm can fairly compete with the others (which do not use parallel computing either), and that the P-TC algorithm may be advantageously compared to all

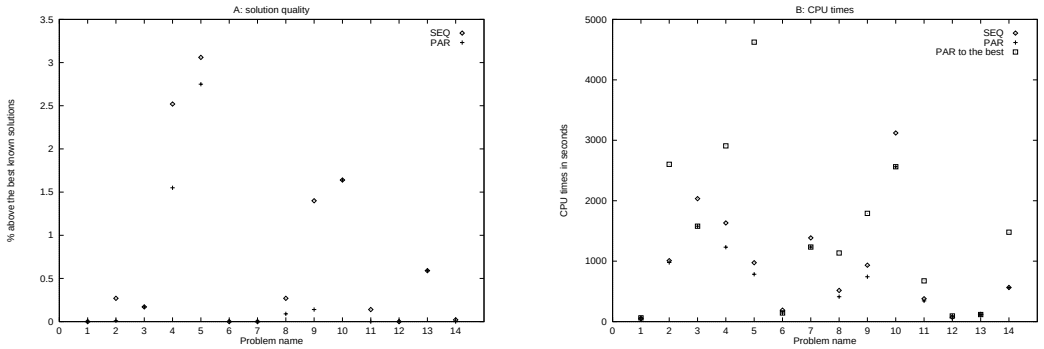

Fig. 6. Solution quality and CPU times for S-TC and P-TC algorithms

these algorithms.

5.3 Analyzing the node-ejection neighborhood search

In order to evaluate the performance of the node ejection chain approach, and better understand the results obtained, we have carefully implemented a 2-interchange mechanism like the one used by the OTS algorithm in Osman [32]. In Figure 7, diagram A shows an example of the relative superiority of the node-ejection chain procedure over the 2-interchange procedure on the algorithm convergence from a randomly generated initial solution for the biggest problem C5 (for which our algorithm obtained worse results). Similarly, diagram B illustrates how the number of submoves (levels) varies throughout the search for the same problem.

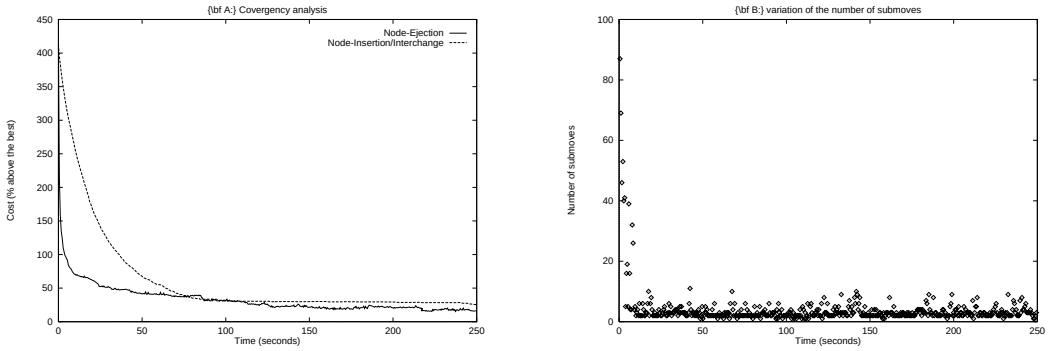

Fig. 7. Convergence analysis and variation in the number of submoves.

6 Conclusion

We have described a new tabu search algorithm for the VRP. It differs from other implementations in the literature in several respects. In particular, we have used compound moves based on an ejection chain process, rather than the

standard k-opt based processes. Furthermore, the algorithm includes a heuristic post-optimization procedure based on the same ejection chain concepts, which improves each route of the VRP separately by solving the corresponding TSP.

Tabu search can be implemented at a variety of levels, to establish different tradeoffs between simplicity of programming and sophistication of the search. We have shown how advanced concepts of the method can be used in ejection chain methods. Indeed, different approaches to *oscillation strategies* which exploit the moves' characteristics have been considered allowing infeasibility regions to be crossed. Thus, an admissible compound move can be obtained by a sequence of inadmissible moves. This is often efficient when no improvement has been found during the intensification and diversification phases, thus proving that it is a powerful strategy to overcome local minima.

In addition, we have described and tested a parallel tabu search algorithm for the VRP. In the neighborhood search the algorithm uses a new concept of creating compound moves based on a node ejection chain process, which has already proved to be efficient for the sequential version of the algorithm.

The parallel implementation was based on a synchronous model and different levels of parallelization were used. Experiments showed that search strategies based on different parameter settings make it possible to explore the solution space more extensively and to find better solutions. We have noticed that the process which finds the best global solution usually changes at each point of synchronization, hence it reflects a dynamic adjustment of parameters which are the most suitable to improve the best global solution at each step.

Indeed, two parallelization techniques were used in order to accelerate the search process. We first discuss the expected gain related to the design of the parallel algorithm. Then we make some comments one the gain obtained with the present implementation.

The time per iteration may be reduced by half, using evaluations of an ejection move and the associated trial move separated into two independent operations. Furthermore, another method is used assigning each route to a distinct process thus accelerating the post-optimization phase to the maximum time required for the reoptimization of a route.

However, with the present parallel system this expected gain was not fully achieved because a virtual parallelism has to necessarily occur due to limitations on the number of processors available.

Finally, the empirical performance of the proposed algorithms was tested for 14 problems taken from the literature which have proved to be very difficult in terms of finding good feasible solutions. Computational results confirm the

very high efficiency of the sequential version algorithm which is increased using the parallel approach.

We think that the effectiveness of the parallel algorithm may be increased if a greater number of processors is used. Nevertheless, as the number of processors grows an asynchronous model may be desirable.

Also, the type of processors is an important feature for the efficiency of the algorithm and therefore an appropriate parallel system should be chosen supporting the different granularities of the parallel tasks. Hence, the post-optimization process holds a "coarse-grain" parallelism when compared with the "fine-grain" required by the move evaluation process.

References

[1] K. Altinkemer and B. Gavish. Parallel savings based heuristic for the delivery problem. *Operations Research*, 39:456–469, 1991.

[2] B. Cao and F. Glover. Tabu search and ejection chains - application to a node weighted version of the cardinality-constrained TSP. *Management Science*, 43(7):908–921, 1997.

[3] J. Chakrapani and J. Skorin-Kapov. Massively parallel tabu search for the quadratic assignment problem. *Annals of Operations Research*, 41:327–341, 1993.

[4] N. Christofides, A. Mingozzi, and P. Toth. *The Vehicle Routing Problem. Combinatorial Optimisation*, chapter 11, pages 315–338. Wiley, Chichester, (N. Christofides, A. Mingozzi, P. Toth and C. Sandi, Eds), 1979.

[5] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[6] T. Crainic, M. Toulouse, and M. Gendreau. Towards a taxonomiy of parallel tabu search algorithms. Technical Report CRT-933, Centre de Recherche sur les Transports, Université de Montréal, 1993.

[7] M. Desrochers and T. Verhoog. A matching based algorithm for the vehicle routing problem. Technical Report Cahier du GERAD G-89-04, Ecole de Haultes Etudes Commerciales de Montréal, 1989.

[8] U. Dorndorf and E. Pesch. Fast clustering algorithms. *ORSA Journal on Computing*, 6:141–153, 1994.

[9] M. L. Fisher. Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42(4):626–642, 1994.

[10] M. L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11:109–124, 1981.

[11] M. Gendreau, A. Hertz, and G. Laporte. Tabu search heuristic for the vehicle routing problem. *Management Science*, 40:1276–1290, 1994.

[12] B. E. Gillett and L. R. Miller. A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, 22:340–349, 1974.

[13] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 5:533–549, 1986.

[14] F. Glover. Tabu search - part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[15] F. Glover. Tabu search - part II. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[16] F. Glover. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, 65:223–253, 1992.

[17] F. Glover. New ejection chain and alternating path methods for traveling salesman problems. *Computer Science and Operations Research*, pages 449–509, 1992.

[18] F. Glover. Tabu search fundamentals and uses. Technical report, Graduate School of Business and Administration, University of Colorado at Boulder, 1995.

[19] F. Glover. Tabu thresholding: Improved search by nonmonotonic trajectories. *ORSA Journal on Computing*, 7:426–442, 1995.

[20] F. Glover. Multilevel tabu search and embedded search neighborhoods for the traveling salesman problem. *ORSA Journal on Computing*, 1996. Forthcoming.

[21] F. Glover and M. Laguna. *Tabu Search*, pages 71–140. Blackwell Scientific Publishing, (C. Reeves, Eds), 1993.

[22] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, MA, 1997.

[23] F. Glover, E. Pesch, and I. Osman. Efficient facility layout planning. *International Journal of Production Research*, 37(2):263–283, 1999.

[24] E. Hadjiconstantinou, N. Christofides, and A. Mingozzi. A new exact algorithm for the vehicle routing problem based on q-path and k-shortest path relaxations. *Annals of Operations Research*, 61:21–43, 1996.

[25] P. Hansen and N. Mladenović. *An Introduction to Variable Neighborhood Search*. Methaheuristics: Advances and Trends to Local Search Paradigms for Optimization. S. Voss, S. Martello, I.H. Osman, C. Roucairol (eds), Kluwer Academic Publishers, 1999.

[26] F. Harche and P. Raghavan. A generalised exchange heuristic for the capacited vehicle routing problem. Technical report, Sterm School of Business, New York University, 1991.

[27] R. Hubscher and F. Glover. Ejection chain methods and tabu search for clustering. Technical report, Graduate School of Business and Administration, University of Colorado at Boulder, 1992.

[28] M. Laguna, J. P. Kelly, J. L. Gonzalez-Valarde, and F. Glover. Tabu search for multilevel generalized assignment problems. *European Journal of Operational Research*, 82:176, 1995.

[29] G. Laporte and I. H. Osman. Routing problems: A bibliography. *Annals of Operations Research*, 61:227–262, 1995.

[30] R. H. Mole and S. R. Jameson. A sequential route-building algorithm employing a generalised savings criterion. *Operational Research Quarterly*, 27:503–511, 1976.

[31] N.Mladenović and P. Hansen. Variable neighborhood decomposition search. *Computers and Operations Research*, 24:1097–1100, 1997.

[32] I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41:421–451, 1993.

[33] C. Rego. *Local Search and Neighborhood Structures for Vehicle Routing Problems: Sequential and Parallel Algorithms*. PhD thesis, PRiSM Laboratory, University of Versailles, 1996.

[34] C. Rego. Relaxed tours and path ejections for the traveling salesman problem. *European Journal of Operational Research*, 106:522–538, 1998.

[35] C. Rego. A subpath ejection method for the vehicle routing problem. *Management Science*, 44(10):1447–1459, 1998.

[36] J. Renaud, F. Boctor, and G. Laporte. An improved petal heuristic for the vehicle routing problem. *Journal of Operational Research Society*, 17:329–336, 1996.

[37] Y. Rochat and E. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1:147–167, 1995.

[38] W. Stewart Jr. and B. Golden. A Lagrangian relaxation heuristic for vehicle routing. *European Journal of Operational Research*, 15:84–88, 1984.

[39] E. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23:661–673, 1993.

[40] J. Xu and J. Kelly. A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation Science*, 30:379–393, 1996.