Heuristic Solutions of Vehicle Routing Problems in Supply Chain Management

Yannis Marinakis (marinakis@ergasya.tuc.gr)
DSS Laboratory
Department of Production Engineering and Management
Technical University of Crete
73100 Chania, Greece

Athanasios Migdalas (sakis@verenike.ergasya.tuc.gr)
DSS Laboratory
Department of Production Engineering and Management
Technical University of Crete
73100 Chania, Greece

Abstract

The distribution of commodities, known by the generic name vehicle routing problem, is one of the most important components of supply chain. The vehicle routing problem, which is a hard combinatorial problem, has therefore attracted considerable research attention and a number of algorithms has been proposed for its solution. In this paper we present an extensive review of heuristic solution techniques for the vehicle routing and traveling salesman problems.

Keywords: Supply Chain Management, Vehicle Routing Problem, Traveling Salesman Problem, Distribution Problem, Combinatorial Optimization, Heuristics

1 Introduction

In this paper we present a review of heuristic algorithms for vehicle routing and traveling salesman problems. We focus on heuristic algorithms because no exact algorithm can be guaranteed to find optimal tours within reasonable computing time when the number of cities is large. This is due to NP–hardness of the problems. The structure of the paper is as follows: In Section 2, we present some basic characteristics of the supply chain management and we focused on the role of transportation problems. In Section 3, we present the traveling salesman problem, the vehicle routing problem and a few extensions of the latter. In Section 4, we describe classic heuristics for the solution of the mentioned problems. In Section 5, we review modern techniques that. Section 6 is devoted to computational comparisons between the reviewed algorithms.

2 Supply Chain Management

A complete logistics [5, 25] system covers the entire process of shipping raw materials and input requirements from suppliers to plants, the conversion of the inputs into products at certain plants, the shipping of the products to various warehouses or depots, and the eventual delivery of these products to the final customers. The distribution activities of a firm comprise all shipping and storage of goods downstream from the plants.

We classify the decisions for supply chain management into two broad categories — strategic and operational. Strategic decisions are made typically over a longer time horizon. On the other hand, operational decisions are short term, and focus on activities over a day–to–day basis. There are four major decision areas in supply chain management, and there are both strategic and operational elements in each of these areas:

1. Location.
2. Production.
3. Inventory.
4. Transportation — Distribution.

2.1 Transportation decisions

Transport decisions [5] can involve mode selection, shipment size, and routing and scheduling. These decisions are influenced by the proximity of warehouses to customers and plants, which, in turn, influence warehouse location. Inventory levels also respond to transport decisions through shipment size.

Transportation often represents the most important single element in logistics costs for most firms. Transportation is a key decision area within the logistics mix. Except for the cost of purchased goods, transportation absorbs, on the average, a higher percentage of logistics costs than any other relevant activity. Because transportation costs typically range between one third and two thirds of total logistics costs, improving efficiency through the maximum utilization of transportation equipment and personnel is of a major interest. The time that goods are transit reflects on the number of shipments that can be made with a vehicle within a given period of time and on the total transportation costs for all shipments. To reduce transportation costs and also to improve customer services finding the best — in terms of time or distance minimization — routes that a vehicle should follow through a network of roads, rail and other shipping lines is frequently an important decision problem.

3 The vehicle routing problem

The *distribution* or *vehicle routing problem* (VRP) is often described as the problem in which vehicles based on a central depot are required to visit geographically dispersed customers in order to fulfill known customer demands. The problem is to construct a low cost, feasible set of routes — one for each vehicle. A route is a sequence of locations that a vehicle must visit along with the indication of the serve it provides [9]. The vehicle must start and finish its tour at the depot.

We can say that the problem arises as a generalization of the traveling salesman problem. The *traveling salesman problem* (TSP) requires the determination of a minimal cost cycle that passes through each node of a given graph exactly once [46]. If costs are symmetric, that is, if the cost of traveling between two locations does not depend on the direction of travel, we have a *symmetric* TSP, otherwise we have an *asymmetric* TSP. The *multiple traveling salesman problem* arises if many salesmen or vehicles in the fleet are to leave from and return to the same depot. There are no restrictions on the number of nodes that each vehicle must visit except that each vehicle must visit at least one node.

The vehicle routing problem has a variety of additional constraints and extensions that are often found in real–world problems. These include the following [16]

- Each vehicle can operate on more than one routes, provided that the total time spent on these routes is less than a given bound T.
- Each customer must be visited within a specific time interval, known as *time window*.
- The problem may involve both deliveries to and collections from customers. In addition, it may be possible to mix deliveries and collections on a single route, or alternatively, it may be required from a vehicle to first perform all the deliveries in the route before performing the collections.

- Vehicles may also be associated with time windows within which they are allowed to operate.

3.1 Variants of the Vehicle Routing Problem

In this section we present some variants of the vehicle routing problem, namely, the multiple depot vehicle routing problem, the multiple commodities vehicle routing problem, the stochastic vehicle routing problem, the vehicle scheduling problem and others.

Many applications [50, 60] involve pickup and delivery services between the depot and peripheral locations (warehouses, stores, stations). 'Delivery' refers to transportation of goods from the depot to customers, and 'pickup' means shipment in the opposite direction (to the depot). In the literature this problem is known as *pick-up and delivery problem* (PDP) or *vehicle routing problem with back-hauls* (VRPB) [63, 65]. The objective is to find a set of vehicles routes that serve the delivery and backhaul customers so that vehicle capacities are not violated and the total distance traveled is minimized.

In companies [8, 16] with more than one depot, it is often the case that each depot is autonomous, with its own fleet of vehicles and its own geographical customer area to serve. In such cases, the company would simply face a number of similar single - depot vehicle routing problems. In other cases depot operations are interdependent and vehicles leaving one depot may, after delivering to customers, end up at another depot. These problems are called *multiple depot vehicle routing problem*

In some cases [16], the vehicles are compartmented so that different commodities are stored in segregated compartments. Each customer may require specific quantities of different types of commodities. The problem is then called *multiple commodities vehicle routing problem.*

The problem of managing routing and dispatch operations under conditions of random demand fluctuations, is often referred to as the *stochastic routing problem* [34]. The stochastic vehicle routing problem is usually formulated under the following conditions [9]:

1. Customer demand is a random variable with a known probability distribution.
2. Routes must be designed before the actual demands become known.
3. The objective is to minimize expected travel distance.

Vehicle scheduling problems can be thought of as routing problems with additional constraints imposed by time periods during which various activities may be carried out

[9]. Three constraints commonly determine the complexity of the vehicle scheduling problems:

- The length of the time that a vehicle may be in operation before it must return to the depot for service or refueling.
- The fact that certain tasks can only be carried out by certain vehicle types.
- The presence of a number of depots where vehicles may be housed.

In the *school bus routing and scheduling problem* [8, 10], there are a number of schools, each having being assigned a set of bus stops, with a given number of students assigned to each stop, and time windows for the pickup and the delivery of the students. The problem is to minimize the number of buses used or total transportation costs while serving all the students and satisfying all the time windows.

In the problem of *routing and scheduling with full loads and time windows*, a set of demands is specified for a number of origin-destination pairs. Each demand is a full trailer which must be loaded onto a tractor at an origin and unloaded at a destination. These stops must satisfy prespecified time window constraints and the goal is to design routes and schedules for the fleet of tractors. In most cases, the objective is to minimize total transportation costs or the number of tractors used.

In the *multi–vehicle covering tour problem* [33] we are given two sets of locations. The first set, V, consists of potential locations at which some vehicles may stop, and the second set, W are locations not actually on the vehicle routes but within an acceptable distance of a vehicle route. The problem is to construct several vehicle routes through a subset of V, all starting and ending at the same locations, subject to some side constraints, having a total minimum length, and such that every location of W is within a reasonable distance of a route.

An important aspect [7] of the vehicle routing problem that has been largely overlooked is the use of *satellite facilities* to replenish vehicles during a route. When possible, satellite replenishment allows the drivers to continue making deliveries until the close of their shift without necessarily returning to the central depot. This situation arises primarily in the distribution of fuel and certain retail items.

The *vehicle routing problem with trailers* [28] is concerned with the case where a vehicle consists of a truck and a trailer. Both can carry goods. The use of truck and trailer may cause problems when serving customers that are located in the center of a city or customers who have little space nearby for maneuvering the vehicle. Time and trouble could be saved if these customers were served by the trucks only.

4 Classic heuristics for the traveling salesman and the vehicle routing problems

In this section we examine three different classes of heuristics — tour construction procedures, two phases algorithms and tour improvement procedures. Tour construction procedures generate an approximately optimal tour for the distance matrix. Two phases algorithms, like cluster first – route second procedures, which during the first phase group or cluster demand nodes and then, during a second phase, design economical routes. Tour improvement procedures attempt to find a better tour given an initial one.

4.1 Constructive methods

4.1.1 Nearest Neighbor Procedure

In this heuristic, the salesman starts at some city and then visits the city nearest to the starting one. From there, he visits the nearest unvisited city, until all cities are visited, and then returns to the starting city [54, 59].

Step 1. Start with any node as the beginning of a path.

Step 2. Find a node, not already on the path, which is closest to the last added node. Add it to the path.

Step 3. Repeat Step 2 until all nodes belong to the path. Then, join the first and the last nodes of the path.

For symmetric, complete graphs, the worst case behavior of the algorithm is

$$\frac{\text{length of nearest neighbor tour}}{\text{length of optimal tour}} \leq \frac{1}{2}\lceil \log_2(n) \rceil + \frac{1}{2},$$

where $\lceil x \rceil$ is the smallest integer $\geq x$, and n is the number of nodes in the network. The nearest neighbor algorithm is of $O(n^2)$ time complexity.

4.1.2 Insertion Procedures

The insertion procedures [30, 59] takes a subtour of k nodes and attempts to determine which node (not already in the subtour) should join the subtour next (the *selection step*) and then determines where in the subtour it should be inserted (the *insertion step*). The most known of these algorithms is the *nearest insertion algorithm*:

Step 1 Start with a subgraph consisting of only one node, say i.

Step 2 Find node k such that c_{ik} is minimal and form the subtour $i - k - i$.

Step 3 (Selection) Given a subtour, find node k, not already in the subtour, closest to any subtour node.

Step 4 (Insertion) Find the arc (i, j) in the subtour which minimizes $c_{ik} + c_{kj} - c_{ij}$. Insert k between i and j.

Step 5 Go to Step 3 unless a Hamiltonian cycle has been formed.

The worst case behavior of the algorithm is

$$\frac{\text{length of nearest insertion tour}}{\text{length of optimal tour}} \leq 2,$$

and its time complexity $O(n^2)$.

Similar to the nearest insertion procedure are the *cheapest insertion* [54], the *arbitrary insertion* [9], the *farthest insertion* [54], the *quick insertion* [9], and the the *convex hull insertion* [9] algorithm.

4.1.3 Clarke and Wright algorithm

This savings algorithm is an exchange procedure [17, 30, 44], that was originally developed for the VRP. It can, however, be applied to the TSP as well:

Step 1. Select any node as the central depot and denote it as node 1.

Step 2. Compute the savings $s_{ij} = c_{1j} + c_{1i} - c_{ij}$ for $i, j = 2, 3, \cdots, n$.

Step 3. Order the savings from largest to smallest.

Step 4. Starting at the top of the savings list and moving downwards, form larger subtours by linking appropriate nodes i and j. Repeat until a tour is formed.

Next the Clarke and Wright algorithm for the solution of the VRP [16] is given:

Step 1. Calculate the savings $s_{ij} = c_{1j} + c_{1i} - c_{ij}$ for all pairs of customers i and j. Note that s_{ij} is the saving in cost that would result if the link (i, j) is made to produce the route $(1, i, j, 1)$ instead of supplying i and j on two routes $(1, i, 1)$ and $(1, j, 1)$.

Step 2. Order the savings in descending order.

Step 3. Starting at the top of the list do the following.

Parallel version

Step 4. If a given link results in a feasible route according to the constraints of the VRP, then append this link to the solution, if not reject the link.

Step 5. Try the next link in the list and repeat Step 4 until no more links can be chosen.

Sequential version

Step 4. Find the first feasible link in the list which can be used to extend one of the two ends of the currently constructed route.

Step 5. If the route can not be expanded further, terminate the route. Choose the first feasible link in the list to start a new route.

Step 6. Repeat Steps 4 and 5 until no more links can be chosen.

The worst case behavior for this approach is bounded by a linear function in $log_2(n)$. The calculation of the matrix s_{ij} in Step 2 requires about cn^2 computations for some constant c. Next, in Step 3 savings can be sorted into nonincreasing order via Heapsort method in a maximum of $cn^2 lg(n)$ computations. Step 4 involves at most n^2 computations. Thus, the Clark and Wright savings procedure requires an order of $n^2 \log_2(n)$ computations.

4.1.4 Nearest Merger algorithm

The nearest merger method [59] when applied to a TSP of n nodes constructs a sequence $S_1, \cdots, S_n$ such that each S_i is a set of $n - i + 1$ disjoint subtours covering all the nodes.

Step 1. S_1 consists of n subtours, each containing a single node.

Step 2. For each $i < n$, find an edge (a_i, b_i) such that $c_{a_i b_i} = \min\{c_{xy}$ for x and y in different subtours in $S_i\}$. Then S_{i+1} is obtained from S_i by merging the subtours containing a_i and b_i.

The worst case behavior of the algorithm is

$$\frac{\text{length of nearest merger tour}}{\text{length of optimal tour}} \leq 2.$$

This approach requires an order of n^2 computations.

4.1.5 Christofides algorithm

Christofides [9, 54] suggested a method of transforming spanning trees to Eulerian graphs. For this, it is sufficient to add a perfect matching on the odd–degree nodes of the tree. After adding the perfect matching edges, all node degrees are even and hence the graph is Eulerian. The following procedure for the solution of the TSP can be based on this:

Step 1. Find a minimal spanning tree T of the given graph G.

Step 2. Identify all the odd degree nodes in T. Solve a minimum cost perfect matching on the odd degree nodes using the original cost matrix. Add the edges from the matching to the edges of T to obtain an Euler cycle. In this subgraph every node is of even degree although some nodes may have degree greater than 2.

Step 3. Remove polygons over the nodes with degree greater than 2 and transform the Euler cycle into a Hamiltonian cycle.

The worst case behavior of the algorithm is

$$\frac{\text{length of Christofides tour}}{\text{length of optimal tour}} \leq 1.5.$$

Since most of the computation time is consumed by the minimum matching subroutine, this heuristic is $O(n^3)$. However, the number of odd nodes may be considerable less than n .

4.2 2–phase algorithms

In two phase methods, the first phase consists of clustering the customers by assigning them to vehicles, and the second phase routes these clusters. The best known algorithms of this category are the sweep algorithm, the Mole and Jameson algorithm and the two phase method of Fisher and Jaikumar

4.2.1 The sweep algorithm

The sweep algorithm was originally devises by Gillet and Miller [9]. This approach constructs a solution in two stages. First, it assigns nodes to vehicles and then it sequences the order in which each vehicle visits the nodes assigned to it. The procedure is as follows [17, 16, 44]:

Phase I

Step 1. Choose an unused vehicle k.

Step 2. Starting from the unrouted customer i with smallest angle ϑ_i, include consecutive customers $i+1, i+2, \cdots$ in the route until the capacity constraint of the vehicle k is reached.

Step 3. If all customers are swept or if all vehicles have been used, go to phase II, else return to Step 1.

Phase II

Step 4. Solve a TSP for every set of customers assigned to a vehicle to form the final routes.

4.2.2 Method of Mole & Jameson

The algorithm of Mole and Jameson is a sequential procedure [17] in which, for a given value of two parameters λ and μ, the following two criteria are used to expand a route under construction:

$$\begin{aligned} e(i,l,j) &= c_{il} + c_{lj} - \mu c_{ij} \\ \sigma(i,l,j) &= \lambda c_{0l} - e(i,l,j) \end{aligned}$$

The algorithm proceeds as follows:

Step 1. For each unrouted customer x_l compute the feasible insertion in the emerging route R as

$$e(i_l, l, j_l) = \min[e(r,l,s)]$$

for all adjacent customers $x_r, x_s \in R$, where x_{i_l} and x_{j_l} are customers between which x_l results in the best insertion.

Step 2. The best customer $x_{l^\star}$ to be inserted in the route is computed as the one for which

$$\sigma(i_{l^\star}, l^\star, j_{l^\star}) = \max[\sigma(i_l, l, j_l)],$$

where x_l is unrouted and feasible.

Step 3. Insert $x_{l^\star}$ in route R between $x_{i_l^\star}$ and $x_{j_l^\star}$.

Step 4. Optimize route R using r–optimal methods (§4.3).

Step 5. Return to Step 1 to start a new route R either until all customers are routed or no more customers can be routed.

4.2.3 Method of Fisher & Jaikumar

The first [22, 23] phase of this algorithm performs a parallel clustering by solving optimally a generalized assignment problem.

Phase I

Step 1. Choose m customers to be seeds of clusters and allocate a vehicle to each.

Step 2. For each customer i and for each cluster k, compute an insertion cost d_{ik} relative to the seed of the cluster.

Step 3. Solve a corresponding generalized assignment problem (GAP) to obtain clusters.

Phase II

Step 4. Solve a TSP for every set of customers in the clusters implied by the solution to GAP.

4.3 Improving Heuristics or Local Search Heuristics

A local search algorithm [1] is built around a neighborhood search procedure. Given a tour the algorithm examines all the tours that are "neighboring" to it and tries to find a shorter one. The definition of "neighbor" varies with the details of the particular local search heuristic [39]. The overall process goes as follows: Starting with some initial tour, chosen arbitrary or generated by some other heuristic, if there is no neighboring tour which is shorter than the original one, the process halts. The initial tour is a local optimum with respect to the chosen neighborhood. Otherwise, a shorter neighbor of the original tour is used as a new starting point, and the process is repeated. The method must eventually terminate as there is only a finite number of possible tours.

A neighborhood N for a problem instance (S, f), where S is the set of feasible points and f the objective function, can be defined as a mapping from S to its power set, i.e., $N : S \to 2^S$. $N(s)$ is called the *neighborhood* of $s \in S$, and contains all the solutions that can be reached from s by a *single move*. Here, the meaning of a move is that of an operator which transforms one solution to another with small modifications. A solution x is called a *local minimum* of f with respect to the neighborhood N if $f(x) \leq f(y), \forall y \in N(x)$.

A local search or improvement algorithm begins with an arbitrary solution and ends up in a local minimum where no further improvement is possible. There are many different ways to implement local search. The best known improvement heuristics for the TSP and VRP are the edge exchange heuristics. The 2–opt and 3–opt heuristics were introduced by Lin [47] and the LK heuristic was introduced by Lin and Kernigham [48]. For a given k, if we define a k–exchange to be the deletion of k edges from a tour and their replacement by k other edges, not already in the tour, then we say that a tour is k–optimal if it is not possible to improve it further via such k–exchanges.

4.3.1 2–opt method

A 2–opt procedure [47, 54], in general, consists of eliminating two edges and reconnecting the two resulting paths in order to obtain a new tour. Note that there is only one way to reconnect the paths. The 2–opt procedure was introduced by Lin (1965) for the TSP:

Step 1. Let T be the current tour.

Step 2. For every node $i = 1, 2, \cdots, n$: Examine all 2–opt moves involving the edge between i and its successor in the tour. If it is possible to decrease the tour length this way, then choose the best such 2–opt move and update T

Step 3. If no improving move could be found, then stop.

In the worst case, it can only be guaranteed that an improving move decreases the tour length by at least one unit. No polynomial worst case bound for the number of iterations to reach a local optimum can be given. Checking whether an improving 2–opt move exists takes $O(n^2)$ time.

4.3.2 3–opt method

The 3–opt heuristic is quite similar the 2–opt but it introduces more flexibility in modifying the current tour, because it uses a larger neighborhood. The tour breaks into three parts instead of only two. There are eight ways to connect the resulting three paths in order to form a tour. There are $\binom{n}{3}$ ways to remove three edges from a tour.

4.3.3 Lin – Kerningham algorithm

This algorithm was developed by Lin and Kernigham (LK) [48, 35] and for many years was considered to be the best heuristic for the TSP. The LK–algorithm decides dynamically at each iteration the number of edges that should be exchange. The procedure works as follows [32, 48]:

Step 1. Generate a random initial solution T.

Step 2.

Step 2.1. Set the iteration counter $i = 1$.

Step 2.2. Select x_i and y_i as the most out of place pair at the i-th iteration. This generally means that x_i and y_i are chosen to maximize the improvement when $x_1, \cdots, x_i$ are exchanged with $y_1, \cdots, y_i$. x_i is chosen from $T - \{x_1, \cdots, x_i\}$ and y_i from $S - T - \{x_1, \cdots, x_i\}$.

Step 2.3. If it appears that no more gain can be made, go to Step 3. Otherwise, set i = i + 1 and go back to Step 2.2.

Step 3. If the best improvement is found for $i = k$, exchange $x_1, \cdots, x_k$ with $y_1, \cdots, y_k$, to obtain a new T, and go to Step 2. If no improvement was found, go to Step 4.

Step 4 (Multistart). Repeat from Step 1 if desired. Otherwise, terminate.

4.3.4 Or–opt

The Or–opt procedure, well known as node exchange heuristic, was first introduce by Or [66]. It removes a sequence of up–to–three adjacent nodes and inserts it at another location within the same route. Or–opt can be considered as a special case of 3–opt (three arcs exchanges) where three arcs are removed and substituted by three other arcs. When removing a chain of consecutive nodes in Or–opt, two arcs are deleted, and the third is eliminated when inserting the chain back into the route. However, the number of possible Or–exchanges is far less than that of possible 3–exchanges. Or–opt is also shown to produce improved solutions of comparable quality to those produced by 3–opt, while requiring significantly less computational time.

Or–opt algorithm can be described as follows [43]:

Step 1. Consider an initial tour and set $t = 1$ and $s = 3$.

Step 2. Remove from the tour a chain of s consecutive vertices, starting with the vertex in position t, and tentatively insert it between all remaining pairs of consecutive vertices on the tour

Step 2.1. If the tentative insertion decreases the cost of the tour, implement it immediately, thus defining a new tour. Set $t = 1$ and repeat Step 2.

Step 2.2. If no tentative insertion decreases the cost of the tour, set $t = t + 1$. If $t = n + 1$ then proceed to Step 3, otherwise repeat Step 2.

Step 3. Set $t = 1$ and $s = s - 1$. If $s > 0$ go to Step 2, otherwise stop.

4.3.5 GENI and GENIUS

The GENI algorithm was presented by Gendreau, Hertz and Laporte (1992) [40]. GENI is a hybrid of tour construction and local optimization. Suppose that $c_1, c_2, \cdots, c_N$ is an arbitrary ordering of the cities. Starting with the partial tour consisting of the first three cities, c_1, c_2, c_3, new cities are added to the current tour in the order given, starting with c_4. To add city c_i, possible ways of inserting it into the tour are considered as well as a 3–opt or 4–opt move with the c_i as the endpoint of one of the deleted edges. The range of possibilities is restricted by requiring that certain of the inserted edges link cities to members of their nearest neighbor lists, where only cities currently in the tour qualify to be in such lists. Also the list lengths are constrained to a maximum p. GENIUS is a true local optimization algorithm based on principles similar to GENI's. Given a starting tour generated by GENI, it cycles through the cities looking for improvements.

5 Metaheuristics for the Traveling Salesman and the Vehicle Routing Problems

5.1 Simulated Annealing

Simulated annealing (SA) belongs [19] to a class of local search algorithms that are known as *threshold algorithms*. These algorithms play a special role within local search for two reasons. First, they appear to be quite successful when applied to a broad range of practical problems. Second, some threshold algorithms such as SA have a stochastic component, which facilitates a theoretical analysis of their asymptotic convergence. The approach of SA originates from theoretical physics, where Monte–Carlo methods are employed to simulate phenomena in statistical mechanics. Its predecessor is the so-called Metropolis filter. This simulation method can be motivated as follows. Consider a huge number of particles of fixed volume at some temperature ϑ. Since the particles move, the system can be in various states. The probability that the system is in a state of certain energy E is given by the Boltzmann distribution $f(E) = \frac{\exp(\frac{-E}{\kappa_B \vartheta})}{z(\vartheta)}$, where $z(\vartheta)$ is a normalization factor and κ_B is the Boltzmann constant. This distribution characterizes the statistical equilibrium of the system at temperature ϑ. In the following we present a simulated annealing procedure for the TSP [54]:

Step 1. Compute an initial tour T and choose an initial temperature $\vartheta > 0$ and a repetition factor r.

Step 2. If the stopping criterion is not satisfied:

- **Step 2.1.** Do the following r times.
  - **Step 2.1.1.** Perform a random modification of the current tour to obtain the tour (T') and let $\Delta = c(T') - c(T)$.
  - **Step 2.1.2.** Compute a random number x, $0 \leq x \leq 1$.
  - **Step 2.1.3.** If $\Delta < 0$ or $x < \exp(\frac{-\Delta}{\vartheta})$.
- **Step 2.2.** Update ϑ and r.

Step 3. Output the current tour T as solution.

5.1.1 Algorithm by Alfa, Heragu and Chen

The method described by Alfa, Heragu and Chen [26] can be viewed as route first – cluster second algorithm. A giant tour is first constructed without considering weights and vehicle capacity, and then it is partitioned into segments of consecutive vertices whose total weight does not exceed a given capacity Q. It is implicitly assumed that the number of vehicles is not fixed a priori. Starting with the initial tour, at iteration t, three edges are randomly selected and removed from the tour, as in classic 3–opt.

5.1.2 Osman's simulated annealing algorithm

The simulated annealing implementation proposed by Osman [26] is substantially more involved in several aspects:

1. It uses a better starting solution (using the Clarke and Wright algorithm).
2. Some parameters of the algorithm are adjusted in a trial phase.
3. Richer solution neighborhoods are explored using λ–interchanges (See §5.2.2.).
4. The cooling scheduling is more sophisticated.

The reader can find other algorithms based on simulated annealing in the articles [11, 12, 41, 64].

5.2 Tabu Search

Tabu search (TS) was introduced by Glover [29]. Computational experience has shown that TS is a well established approximation technique [36], which can compete with almost all known techniques and which, by its flexibility, can beat many classic procedures. TS combines [1] the deterministic iterative improvement algorithm with a possibility to accept cost–increasing solutions. Thus, the search is directed away from local minima so that other parts of the search space can be explored. The next solution visited is always chosen to be a legal neighbor of the current solution with the best cost, even if that cost is worse than that of the current solution. The set of legal neighbors is restricted by a tabu list designed to prevent the search from cycling. The tabu list is dynamically updated during the execution of the algorithm. The tabu defines solutions that are not acceptable in the next few iterations. However, a solution in the tabu list may be accepted if its quality is in some sense good enough, in which case it is said to attain a certain aspiration level. A general description of TS is the following:

Step 1. Choose an initial solution x. Set $x^* = x$ (the best solution so far) and $k = 0$

Step 2. Set $k = k + 1$ and generate a subset V^* of solutions in $N(x, k)$, the neighborhood of x at iteration k.

Step 3. Choose a best $y \in V^*$ with respect to the objective function f or some modified objective criterion f^1 and set $x = y$.

Step 4. If $f(x) < f(x^*)$, then set $x^* = x$.

Step 5. If a stopping condition is met, then stop. Else go to Step 2.

5.2.1 Willard's algorithm

One of the first attempts to apply tabu search to the VRP is due to Willard [26]. Here the solution is first transformed into a giant tour by replication of the depot, and neighborhoods are defined as all feasible solutions that can be reached from the current solution using 2–opt or 3–opt exchanges. The next solution is determined by the best non–tabu move.

5.2.2 Osman's algorithm

Osman [26] again defines neighborhoods using a λ–interchange scheme. This includes a combination of 2–opt moves, vertex reassignments to different routes, and vertex interchanges between two routes. In one version of the algorithm, called *best–admissible*, the whole neighborhood is explored and the best non–tabu feasible move is selected. In an other version, called *first–best–admissible*, the first admissible improving move is selected if one exists.

5.2.3 TABUROUTE

The TABUROUTE [26] algorithm involves several innovative features:

- The neighborhood structure is defined by all solutions that can be reached from the current solution by removing a vertex from it and inserting it into another route in its neighborhood using GENI (see section 4.3.5).
- The search procedure examines solutions that may be infeasible with respect to the capacity or maximum route length constraints.
- TABUROUTE does not actually use a tabu list, but random tabu tags.
- TABUROUTE uses a diversification strategy. This is achieved by penalizing vertices that have been moved frequently in order to increase the probability of considering slow–moving vertices.

5.2.4 Taillard's algorithm

The Tailard tabu search [26] implementation contains some of the features of the algorithm presented in section 5.2.3, namely, random tabu durations and diversification. It defines neighborhoods using λ–interchanges like the algorithm presented in section 5.2.2. Rather than executing insertions with GENI, the algorithm uses standard insertions.

5.2.5 The Xu and Kelly algorithm

Xu and Kelly [45] used a more sophisticated neighborhood structure. They consider swaps of vertices between two routes, a global repositioning of some vertices into other routes, and local route improvements. The global repositioning strategy solves a network flow model to optimally relocate given numbers of vertices into different routes. Route re-optimizations are performed by means of 3–opt exchanges and a TS improvement routine. The algorithm is governed by several parameters which are dynamically adjusted through the search. A pool of best solutions are memorized and periodically used to re–initiate the search with new parameter values.

5.2.6 Adaptive Memory Procedure

One of the most interesting developments to have occurred in the area of TS in recent years is the concept of Adaptive Memory developed by Rochat and Taillard [31, 45, 57]. It is mostly used in TS, but its applicability is not limited to this type of metaheuristic. An adaptive memory is a pool of good solutions that is dynamically updated throughout the search process. Periodically, some elements of these solutions are extracted from the pool and combined differently to produce new good solutions. In the VRP, vehicle routes selected from several solutions will be used as a starting point. The extraction process gives a larger weight to those routes belonging to the best solutions.

5.2.7 Other algorithms based on tabu search

Many algorithms based on TS have been proposed during the last five years for the solution of VRPs. Gendreau, Laporte Musaraganyi and Taillard proposed an algorithm [27] for the solution of the heterogeneous vehicle routing problem. Firstly, the algorithm makes use of GENIUS (section 4.3.5), secondly it uses TS, and then it is itself embedded within a so-called Adaptive Memory Procedure (section 5.2.6). Renaud, Laporte and Boctor proposed an algorithm [55] for the solution of the multi-depot vehicle routing problem. The algorithm consists of three phases which are: fast improvement, intensification and diversification. The reader can find other algorithms based on tabu search in the articles [3, 24, 12, 14, 52].

5.3 Genetic Algorithms

This search strategy uses concepts from population genetics and evolution theory to construct algorithms [1, 51, 62] that try to optimize the fitness of a population of elements through recombination and mutation of their genes. The general idea of *genetic local search* is give by the following procedure:

Step 1 (Initialize): Construct an initial population of n solutions.

Step 2 (Improve): Use local search to replace the n solutions in the population by n local optima.

Step 3 (Recombine): Augment the population by adding m offspring solutions, the population size now equals $n + m$.

Step 4 (Improve): Use local search to replace the m offspring solutions by m local optima.

Step 5 (Select): Reduce the population to its original size by selecting n solutions from the current population.

Step 6 (Evolute): Repeat Step 3 through 5 until a stop criterion is satisfied.

5.3.1 GIDEON

GIDEON [26, 61] is a genetic algorithm for VRPs with time windows and capacity constraints, based on a cluster first – route second strategy. The genetic algorithm is only applied during the clustering phase: a procedure, called *genetic sectoring*, partitions the vertices into sector or clusters centered at the depot, as in the sweep algorithm (section 4.2.1).

5.3.2 GENEROUS

GENEROUS [26] is a genetic algorithm for VRPs with time windows. This algorithm avoids the difficulties related to the encoding of a solution into a chromosome by applying the crossover and mutation operators on the solutions themselves. In this algorithm, a new solution is created from two parent solutions 1 and 2 by linking the first customers on a route of parent 1 to the last customers on a route of parent 2, as in the 2–opt exchange procedure. The new route replaces the old one in parent solution 1. A second offspring solution can be created by inverting the roles of the parents solutions.

5.4 Neural Nets Algorithms

The use of artificial neural networks to find good solutions to combinatorial optimization problems has recently caught some attention. A neural network consists of a network [1, 15] of elementary nodes (neurons) that are linked through weighted connections. The nodes represent computational units, which are capable of performing a simple computation, consisting of a summation of the weighted inputs, followed by the addition of a constant called the threshold or bias, and the application of a nonlinear response function. The result of the computation of a unit constitutes its output. This output is used as an input for the nodes to which it is linked through an outgoing connection. The overall task of the network is to achieve a certain network configuration, for instance a required input–output relation, by means of the collective computation of the nodes. This process is often called *self–organization*.

5.4.1 Hopfield Neural Nets

The first attempt to produce a solution for the TSP with neural nets was based on a Hopfield Neural Network [2]. Hopfield networks can be used as associative memories for information storage and retrieval, and to solve combinatorial problems. They belong to the class of recurrent neural networks, that is, outputs of a neural networks are fed back to inputs of previous layers of the network. The implementation of a Hopfield Neural Network requires:

- A transformation of the cost and the constraints of the problem into one function, known as Hopfield energy function.
- The determination of Lagrange parameters.

The above steps are often the key decisive factors on the success of the application. These, are also, the reasons why researchers have difficulty in duplicating Hopfield's result for solving the traveling salesman problem. To map the traveling salesman problem onto the Hopfield framework, a scheme is needed to represent the final state of the network as a tour list. Hopfield and Tank [2] adopted a representation scheme in which the precedence (sequence) of cities in a tour list is encoded by the final states of a set of neurons. For example, for an n–city problem, the network needs n^2 neurons — one neuron for each possible tour position for each city. Since there are n cities, each with n possible tour positions, $n \times n$ neurons are needed.

5.4.2 Self–Organized Maps

Self organized maps are instances of so-called competitive neural networks models [26, 49]. They self–organize by gradually adjusting the weights on their connections.

5.5 Ant Algorithms

The ant system, introduced by Colorni, Dorigo and Maniezzo [13, 18], is a new distributed metaheuristic for hard combinatorial optimization problems and was first used on the traveling salesman problem. Observations on real ants searching for food were the inspiration to imitate the behavior of ant colonies. Real ants are able to communicate information concerning food sources via an aromatic essence, called pheromone. They mark the path they walk on by laying down pheromone in a quantity that depends on the length of the path and the quality of the discovered food source. Other ants can observe the pheromone trail and are attracted to follow it.

The described behavior of real ant colonies can be used to solve combinatorial optimization problems by simulation: artificial ants searching the solution space simulate real ants searching their environment, the objective values correspond to the quality of the food sources and an adaptive memory corresponds to the pheromone trails. In addition, the artificial ants are equipped with a local heuristic function to guide their search through the set of feasible solutions.

5.6 Iterated Lin–Kernigham algorithm

The Iterated Lin Kernigham (ILK) [67, 53] has been proposed by Johnson [40] and it is considered to be one of the best for the TSP. ILK uses LK to obtain a first local minimum. To improve this local minimum, the algorithm examines other local minimum tours 'near' the current local minimum. To generate these tours, ILK first applies a random and unbiased nonsequential 4–opt exchange to the current local minimum and then optimizes this 4–opt neighbor using the LK algorithm. If the tour obtained by this process is better than the current local minimum then Iterated LK makes this tour the current local minimum and continues from there using the same neighbor generation process. Otherwise, the current local minimum remains as it is and further random 4–opt moves are tried. The algorithm stops when a stopping criterion based either on the number of iterations or the computational time is satisfied.

The random 4–opt exchange performed by ILK is called *double–bridge move* and plays a diversification role for the search process; It tries to propel the algorithm into a different area of the search space while preserving at the same time large parts of the structure of the current local minimum. The ILK procedure is as follows:

Step 1. Generate a random tour T.

Step 2. Do the following for some prespecified number, M, of iterations:

- **Step 2.1.** Perform an (unbiased) random 4–opt move on T, obtaining T^1.
- **Step 2.2.** Run LK on T^1, obtaining T^2
- **Step 2.3.** If $length(T^2) \leq length(T)$, set $T = T^2$

Step 3 Return T.

5.7 Guided local search

Guided local search (GLS) originally proposed by Voudouris and Chang [67, 4] is a general optimization technique suitable for a wide range of combinatorial optimization problems. The main focus is on the exploitation of problem and search–related information to effectively guide local search heuristics in the vast search spaces of NP-hard optimization problems. This is achieved by augmenting the objective function of the problem to be minimized with a set of penalty terms which are dynamically manipulated during the search process to steer the heuristic to be guided.

GLS augments the cost function of the problem to include a set of penalty terms and passes this, instead of the original one, for minimization by the local search procedure. Local search is confined by the penalty terms and focuses attention on promising regions of the search space. Iterative calls are made to local search. Each time local search gets caught in a local minimum, the penalties are modified and local search is called again to minimize the modification cost function.

5.8 Fast local search

Fast local search (FLS) originally proposed by Voudouris and Chang [67]. FLS works as follows. The current neighborhood is broken down into a number of small sub-neighborhoods and an activation bit is attached to each one of them. The idea is to scan continuously the sub-neighborhoods in a given order, searching only those with the activation bit is set to 1. These sub-neighborhoods are called active sub-neighborhoods. Sub-neighborhoods with the bit set to 0 are called inactive sub-neighborhoods and they are not being searched. The neighborhood search process does not restart whenever we find a better solution but it continues with the next sub-neighborhoods in the given order. This order may be static or dynamic.

5.9 GRASP

The Greedy Randomized Adaptive Search Procedure (GRASP) [20, 21, 38, 37, 56] is a two phase local search. This randomized technique provides a feasible solution within every iteration. The final result is simply the best solution found over all iteration (multi-start local search). Each iteration consists of two phases, a construction phase and a local search procedure. In the construction phase a randomized greedy function is used to build up an initial solution. This solution is then exposed for improvement attempts in the local search phase.

The construction phase can be described as stepwise adding one element at a time to the partial (incomplete) solution. The choice of the next element to be added is determined by ordering all elements in a candidate list with respect to a greedy function. The heuristic is adaptive because the benefits associated with every element are updated at each iteration of the construction phase to reflect the changes brought on by the selection of the previous element. The probabilistic component of a GRASP is characterized by randomly choosing one of the best candidate in the list but not necessary the top candidate.

A generic GRASP algorithm is given below:

Step 1. Input the problem instance.

Step 2. Do until some GRASP stopping criterion is satisfied:

Step 2.1. Execute the construction phase of GRASP in order to obtain a greedy random solution.

Step 2.2. Apply the local search phase to the obtained greedy solution.

Step 2.3. Update the best solution if needed.

Step 3. Return the best solution found.

Although there are many GRASP applications to combinatorial optimization the only known application in the context of the vehicle routing problem is that of Kontoravdis et al. ([6, 42]). Kontoravdis and Bard addressed the problem of finding the minimum number of vehicles needed to serve n customers subject to time windows and capacity constraints:

Step 1. Find a lower bound u on the number of routes needed.

Step 2. Select u seed customers to form a set of initial routes.

Step 3. Calculate the cost of inserting each customers into these routes.

Step 4. Calculate penalties for each insertion.

Step 5. Construct a list L that contains the r largest penalties. From L, randomly choose one customer.

Step 6. Try to insert the customer into the corresponding route. If successful, then go to Step 7. If insertion leads to a time or capacity violation, start a new route between the depot and the customer and go to Step 4.

Step 7. If every customer is routed, then stop, otherwise update the costs and go to Step 5.

Step 8. Repeat Steps 1 to 7 a predetermined number, M,of times and save the best results. During these M iterations, run a post-processor every N ($N \leq M$) times and keep the best results.

6 Computational Results

In order to compare the various algorithms we report the best solution values obtained using each algorithm on the 14 benchmark problems described by Christofides et al. [17]. The algorithms for which we have comparable results are:

- A. The Clarke and Wright algorithm [17].
- B. The Mole and Jameson algorithm [17].
- C. The sweep algorithm [17].
- D. The Fisher & Jaikumar algorithm [45].
- TA. Willard's tabu search algorithm [26].
- TB. Osman's tabu search algorithm — Best admissible strategy [26].
- TC. Osman's tabu search algorithm — First best admissible strategy [26].
- TD. TABUROUTE [26].
- TE. Taillard's tabu search algorithm [26].
- TF. The Xu & Kelly tabu search algorithm [45].
- SAO. Osman's simulated annealing algorithm [45].

In Table 1 the numbers of each box give the total mileage. The best known solutions are indicated with an asterisk. The computational results suggest that the best known solutions for these bench mark problems are always found using Taillard's algorithm, followed closely by the Xu and Kelly algorithm and TABUROUTE. The classic heuristic algorithms give sufficiently good solutions but none of them is competitive with the tabu search implementations. SA implementations produce good quality results, although not so good as the results obtained by Taillard's algorithm.

Concerning genetic algorithms, GIDEON minimizes the total distance and GENEROUS minimizes the total route time [26]. GENEROUS is computationally more expensive than GIDEON as it handles a population of solutions rather than a population of sector angles. For both algorithms the results are less favorable, when compared to TS algorithms.

Besides the 14 benchmark problems by Christofides et al. [17], a large number of problem instances are in the TSPLIB library (`www.elib.zib.de`) and others may be found from the VRP site (`www.geocities.com/ResearchTriangle/7279/vrp.html`).

Problem	A	B	C	D	TA	TB	TC	TD	TE	TF	SAO
1	585	575	532	524⋆	588	524.61⋆	524.61⋆	524.61⋆	524.61⋆	524.61⋆	528
2	900	910	874	857	893	844	844	835.77	835.26⋆	835.26⋆	838.62
3	886	882	851	833	906	835	838	829.45	826.14⋆	826.14⋆	829.18
4	1204	1259	1079	1014⋆	–	1052	1044.35	1036.16	1028.42	1029.56	1058
5	1540	1545	1389	1420	–	1354	1334.55	1322.65	1298.79	1298.58⋆	1378
6	619	599	560	560	–	555.43⋆	555.43⋆	555.43⋆	555.43⋆	555.43⋆	555.43⋆
7	976	969	933	916	–	913	911	913.23	909.68⋆	965.62	909.68⋆
8	973	999	888	885	–	866.75	878	865.94⋆	865.94⋆	881.38	866.75
9	1426	1289	1230	1230	–	1188	1184	1177.76	1162.89⋆	-	1164.12
10	1800	1770	1518	1518	–	1422	1441	1418.51	1397.94⋆	1439.29	1417.85
11	1079	1100	1266	–	–	1042.11⋆	1043	1073.47	1042.11⋆	1042.11⋆	1176
12	831	879	937	824	–	819.59	819.59	819.56⋆	819.56⋆	819.56⋆	826
13	1634	1590	1776	–	–	1547	1547	1573.81	1541.14⋆	1618.55	1545.98
14	877	883	949	876	–	866.37⋆	866.37⋆	866.37⋆	866.37⋆	915.24	890

Table 1. Computational comparison of algorithms.

References

[1] E. Aarts and J.K. Lenstra. *Local Search in Combinatorial Optimization*. Wiley and Sons, 1997.

[2] N. Ansari and E. Hou. *Computational intelligence for optimization*. Kluwer Academic Publishers, first edition, 1997.

[3] P. Augerat, J.M. Belenguer, E. Benavent, A. Corberan, and D. Nannef. Separating capacity constraints in the cvrp using tabu search. *European Journal of Operational Research*, 106:546–557, 1998.

[4] B.D. Baker, V. Furnon, P. Shaw, P. Kilby, and P. Prosser. Solving vehicle routing problems using constraint programming and metaheuristics. *Journal of Heuristics*, 6:501–523, 2000.

[5] R. Ballou. *Business Logistics Management, Planning, Organizing and Controlling the Suply Chain*. Prentice-Hall International, Inc., fourth edition, 1999.

[6] J. Bard, L.Huang, P. Jaillet, and M. Dror. A decomposition approach to the inventory routing problem with satelite facilities. *Transportation science*, 32(2):189–203, 1998.

[7] J.F. Bard, L. Huang, M. Dror, and P. Jaillet. A branch and cut algorithm for the vrp with satelite facilities. *IIE Transactions*, 30:821–834, 1998.

[8] L.D. Bodin and B.L. Golden. Classification in vehicle routing and scheduling. In B. Golden and L. Bodin, editors, *Proceedings of the International Workshop on Current and Future Directions in the Routing and Scheduling of Vehicles and Crews*, pages 97–108. Wiley and Sons, 1979.

[9] L.D. Bodin, B.L. Golden, A.A. Assad, and Michael O. Ball. Routing and scheduling of vehicles and crews. *Computers and Operations Research*, 10:63–211, 1983.

[10] J. Braca, J. Bramel, B. Posner, and D. Simchi Levi. A computerized approach to the new york city school bus routing problem. Technical report, Columbia University, 1994.

[11] A. Van Breedam. Improvement heuristics for the vehicle routing problem based on simulated annealing. *European Journal of Operational Research*, 86:480–490, 1995.

[12] A. Van Breedam. Comparing descent heuristics and metaheuristics for the vehicle routing problem. *Computers and Operations Research*, 28:289–315, 2001.

[13] B. Bullnheimer, R. Hartl, and C. Strauss. An improved ant system algorithm for the vehicle routing problem. pages 1–11, 1997. preprint.

[14] V. Campos and E. Mota. Heuristics procedures for the capacitaded vehicle routing problem. *Computational Optimization and Applications*, 16:265–277, 2000.

[15] Bo Sodererg Carsten Peterson. Artificial neural networks. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 173–214. Wiley and Sons, 1997.

[16] N. Christofides. Vehicle routing. In E.L. Lawer, J.K. Lenstra, A.H.G. Rinnoy Kan, and D.B. Shmoys, editors, *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*, pages 431–448. Wiley and Sons, 1985.

[17] N. Christofides, A. Mignozzi, and P. Toth. The vehicle routing problem. In N. Christofides, editor, *Combinatorial Optimization*, pages 315–338. Wiley and Sons, 1979.

[18] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems*, 26:1–13, 1996.

[19] Peter J.M. van Laarhoven Emile H.L. Aarts, Jan H. M. Korst. Simulated annealing. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 91–120. Wiley and Sons, 1997.

[20] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedure. *Journal of Global Optimization*, 6:109–133, 1995.

[21] P. Festa and M.G.C. Resende. Grasp: An annotated bibliography. Look again, 2000.

[22] M. Fisher. The langrangean relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

[23] M.L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. In B. Golden and L. Bodin, editors, *Proceedings of the International Workshop on Current and Future Directions in the Routing and Scheduling of Vehicles and Crews*, pages 109–124. Wiley and Sons, 1979.

[24] D. Ozgur G. Barbarosoglu. A tabu search algorithm for the vehicle routing problem. *Computers and Operations Research*, 26:255–270, 1999.

[25] R. Ganeshan. An introduction to supply chain management. pages 1–7, 2001.

[26] M. Gendrau, G. Laporte, and J.Y. Potvin. Vehicle routing: Modern heuristics. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 311–336. Wiley and Sons, 1997.

[27] M. Gendreau, G. Laporte, C. Musaraganyi, and E.D. Taillard. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers and Operations Research*, 26:1153–1173, 1999.

[28] J.H. Gerdessen. Vehicle routing problem with trailers. *European Journal of Operational Research*, 93:135–147, 1996.

[29] F. Glover. Tabu search: A tutorial. pages 1–47, 1989.

[30] B.L. Golden. Introduction to and recent advances in vehicle routing methods. In M. Florian, editor, *Transportation Planning Models*, pages 383–418. Elsevier Science Publishers, 1991.

[31] B.L. Golden, G. Laporte, and E.D. Taillard. An adaptive memory heuristic for a class of vehicle routing problems with minmax objective. *Computers and Operations Research*, 24:445–452, 1997.

[32] B.L. Golden and W. Stewart. Empirical analysis of heuristic. In E.L. Lawer, J.K. Lenstra, A.H.G. Rinnoy Kan, and D.B. Shmoys, editors, *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*, pages 207–250. Wiley and Sons, 1985.

[33] M. Hachicha, M.J. Hodgson, G. Laporte, and Frederic Semet. Heuristics for the multi-vehicle covering tour problem. *Computers and Operations research*, 27:29–42, 2000.

[34] M. Haughton and A. Stenger. Semi-variable delivery routes and the efficiency of outbound logistics. *International Journal of Physical Distribution and Logistics Management*, 27:459–474, 1997.

[35] K. Helsgaum. An effective implementation of the lin-kernigham travelling salesman heuristic. *European Journal of Operational Research*, 126:106–130, 2000.

[36] Alan Hertz, Eric Taillard, and Dominique de Werra. Tabu search. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 121–136. Wiley and Sons, 1997.

[37] K. Holmqvist, A. Migdalas, and P.M. Pardalos. Parallel continuous non-convex optimization. In A. Migdalas, P.M. Pardalos, and S. Storøy, editors, *Parallel Computing in Optimization*, pages 471–528. Kluwer Academic Publishers, 1997.

[38] K. Holmqvist, A. Migdalas, and P.M. Pardalos. Parallelized heuristics for combinatorial search. In A. Migdalas, P.M. Pardalos, and S. Storøy, editors, *Parallel Computing in Optimization*, pages 269–294. Kluwer Academic Publishers, 1997.

[39] D. Johnson and C. Papadimitriou. Performance guarantees of heuristic. In E.L. Lawer, J.K. Lenstra, A.H.G. Rinnoy Kan, and D.B. Shmoys, editors, *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*, pages 145–180. Wiley and Sons, 1985.

[40] D.S. Johnson and L.A. McGeoch. The travelling salesman problem: A case study. In E. Aarts and Lenstra J.K, editors, *Local Search in Combinatorial Optimization*, pages 215–310. Wiley and Sons, 1997.

[41] H. Kokubugata, H. Itoyama, and H. Kawashima. Vehicle routing methods for city logistics operations. In *IFAC Transportation Systems*, 1997.

[42] G. Kontoravdis and J.F. Bard. A grasp for the vehicle routing problem with time windows. *ORSA Journal on Computing*, 7(1):10–23, 1995.

[43] G. Laporte. The travelling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59:231–247, 1992.

[44] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59:345–358, 1992.

[45] G. Laporte, M. Gendreau, J.Y. Potvin, and F. Semet. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, 7:285–300, 2000.

[46] E.L. Lawer, J.K. Lenstra, A.H.G. Rinnoy Kan, and D.B. Shmoys. *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization.* Wiley and Sons, 1985.

[47] S. Lin. Computer solutions of the travelling salesman problem. *Bell System Technical Journal*, 44:2245–2269, 1965.

[48] S. Lin and B.W. Kernigham. An effective implementation for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[49] A. Modares, S. Somhom, and T. Enwaka. A self - organizing neural network approach for multiple travelling salesman and vehicle routing problems. *International Transactions in Operational Research*, 6:591–606, 1999.

[50] G. Mosheiov. Vehicle routing with pickup and delivery: Tour - partitioning heuristics. *Computers and Industrial Engineering*, 34:669–684, 1998.

[51] Heinz Muhlenbein. Genetic algorithms. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 137–172. Wiley and Sons, 1997.

[52] W.P. Nanry and J.W. Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B*, 34:107–121, 2000.

[53] David Neto. *Efficient cluster compensation for Lin - Kernigham heuristics*. PhD thesis, Computer science university of Toronto, 1999.

[54] G. Reinelt. *The Travelling Salesman Problem, Computational solutions for TSP Applications*. Springer-Verlag, 1994.

[55] J. Renaud, G. Laporte, and F. Boctor. A tabu search heuristic for the multi - depot vehicle routing problem. *Computers and Operations Research*, 23:229–235, 1996.

[56] M.G.C. Resende. Greedy randomized adaptive search procedure. *Technical report*, 1998.

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

[58] P. Rodriguez, M. Nusbaum, R. Baeza, G. Leon, M. Sepulveda, and A. Cobian. Using global search heuristics for the capacity vehicle routing problem. *Computers and Operations Research*, 25(5):407–417, 1998.

[59] D.J. Rosenkratz, R.E. Stearns, and P.M. Lewis. An analysis of several heuristics for the travelling salesman problem. *SIAM Journal on Computing*, 6:563–581, 1977.

[60] K.S. Ruland and E.Y. Rodin. The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers Mathematical Applications*, 33(12):1–13, 1997.

[61] S. Thangiah. Vehicle routing with time windows using genetic algorithms. *Technical Report*, pages 1–23, 1993.

[62] S.R. Thangiah, I. Osman, T. Sung, and R. Vinayagamoorthy. Algorithms for the vehicle routing problems with deadlines. *American Journal of Mathematical and Management Science*, 13:323–355, 1995.

[63] S.R. Thangiah, J.Y. Potvin, and T. Sung. Heuristics approaches to vehicle routing with backhauls and time windows. *Computers and Operations Research*, 23:1043–1057, 1996.

[64] P. Tian, J. Ma, and D.M. Zhang. Application of the simulated annealing algorithm to the combinatorial optimisation problem with permutation property: An investigation of generation mechanism. *European Journal of Operational Research*, 118:81–94, 1999.

[65] P. Toth and D. Vigo. An exact algorithm for the vehicle routing problem with backhauls. *Transportation Science*, 31(4):372–385, 1997.

[66] D.V. Tung and A. Pinnoi. Vehicle routing-scheduling for waste collection in hanoi. *European Journal of Operational Research*, 125:449–468, 2000.

[67] C. Voudouris and E. Tsang. Guided local search and its application to the travelling salesman problem. *European Journal of Operational Research*, 113:469–499, 1999.