Local Search methods for Vehicle Routing problems

Michel Gendreau

Département d'informatique et de recherche opérationnelle
and
Interuniversity Research Centre on Enterprise Networks,
Logistics and Transportation (CIRRELT)
Université de Montréal

University of Bologna
June 5, 2008
PRESENTATION OUTLINE

1. Introduction
2. Neighbourhoods and search spaces
3. Main classes of local search based search methods
4. Tabu Search
5. Recent trends in Tabu and Local Search
6. Local search operators in routing
7. Local search methods for the CVRP
8. Local search methods for the VRPTW
9. References
INTRODUCTION

• “Tough” combinatorial problems have been around for a long time and some have attracted a lot of interest (e.g.: Traveling Salesman Problem)

• Early 70's: complexity theory

  → NP-hard problems

  ↓

  Little hope of solving efficiently many important problems

  ↓

  What can be done in practical contexts when solutions are needed?

  ↓

  USE HEURISTIC TECHNIQUES

  • constructive heuristics (e.g. “greedy”)

  • iterative improvement methods
CLASSICAL LOCAL IMPROVEMENT HEURISTICS

Key idea:

• In most combinatorial problems, one would expect good solutions to share similar structures.

• Indeed, the best solutions should be obtainable by slightly modifying good ones, and so on…

THUS:

• Start with a (feasible) initial solution.

• Apply a sequence of local modifications to the current solution as long as these produce improvements in the value of the objective function (monotone evolution of the objective).

These methods are the basic (and earlier) trajectory based search methods.

They are usually called “local search” or “neighbourhood search” methods.
PROBLEMS AND LIMITATIONS

- These methods stop when they encounter a local optimum (w.r.t. to the allowed modifications).

- Solution quality (and CPU times) depends on the “richness” of the set of transformations considered at each iteration of the heuristic.

- Another key factor is the definition of the set of solutions explored by the algorithm.
SEARCH SPACES
AND
NEIGHBOURHOODS
SEARCH SPACES

- Simply the space of all possible solutions that can be considered (visited) during the search.

- Could be the set of all feasible solutions to the problem at hand, with each point in the search space corresponding to a solution satisfying all the specified constraints.

- While this definition of the search space might seem quite natural and straightforward, it is not so in many settings, as we shall see later in a few illustrative examples.
NEIGHBOURHOODS

• At each iteration of LS, the local transformations that can be applied to the current solution, denoted $S$, define a set of neighbouring solutions in the search space, denoted $N(S)$ (the neighbourhood of $S$).

• $N(S) = \{\text{solutions obtained by applying a single local modification to } S\}$.

• In general, for any specific problem at hand, there are many more possible (and even, attractive) neighbourhood structures than search space definitions.
EXAMPLES OF SEARCH SPACES AND NEIGHBOURHOODS

Two illustrative problems:

- Vehicle routing problem (VRP)
- Capacitated plant location problem (CPLP)
CLASSICAL VEHICLE ROUTING PROBLEM

- $G = (V, A)$, a graph.
- One of the vertices represents the depot.
- The other vertices customers that need to be serviced.
- With each customer vertex $v_i$ are associated a demand $q_i$ and a service time $t_i$.
- With each arc $(v_i, v_j)$ of $A$ are associated a cost $c_{ij}$ and a travel time $t_{ij}$.
- $m$ identical vehicles of capacity $Q$ are based at the depot.

The CVRP consists in finding a set of routes such that:

- Each route begins and ends at the depot;
- Each customer is visited exactly once by exactly one route;
- The total demand of the customers assigned to each route does not exceed $Q$;
- The total duration of each route (including travel and service times) does not exceed a specified value $L$;
- The total cost of the routes is minimized.
SEARCH SPACES AND NEIGHBOURHOODS FOR THE CVRP

Search space:

- Set of feasible routes.
- Allow routes with capacity violations.
- Allow routes with duration violations.

Neighbourhoods:

- Moving a single customer from its route.
- Insertion can be performed simply or in a complex fashion (e.g., GENI insertions).
- Swap customers.
- Simultaneous movement of customers to different routes and swapping of customers between routes ($\lambda$-interchange of Osman 1993).
- Coordinated movements of customers from one route to another (ejection chains).
- Swapping of sequences of several customers between routes (Cross-exchange of Taillard et al. 1997).
CAPACITATED PLANT LOCATION PROBLEM (CPLP)

• Set of customers \( I \) with demands \( d_i, i \in I \).

• Set \( J \) of “potential sites” for plants.

• For each site \( j \in J \), the fixed cost of “opening” the plant at \( j \) is \( f_j \) and its capacity is \( K_j \).

• \( c_{ij} \): cost of transporting one unit of the product from site \( j \) to customer \( i \).

The objective is to minimize the total cost, i.e., the sum of the fixed costs for open plants and the transportation costs.
CPLP: MATHEMATICAL FORMULATION

\[ (CPLP) \quad \text{Minimize} \quad z = \sum_{j \in J} f_j y_j + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \]

subject to \[ \sum_{j \in J} x_{ij} = d_i, \quad i \in I \]

\[ \sum_{i \in I} x_{ij} \leq K_j y_j, \quad j \in J \]

\[ x_{ij} \geq 0, \quad i \in I, \quad j \in J \]

\[ y_j \in \{0,1\}, \quad j \in J \]

Formulation variables:

- \( x_{ij} \ (i \in I, j \in J) \): quantity shipped from site \( j \) to customer \( i \)
- \( y_j \ (j \in J) \): 0-1 variable indicating whether or not the plant at site \( j \) is open or closed.
Remark 1. For any vector $\tilde{y}$ of location variables, optimal (w.r.t. to this plant configuration) values for the flow variables $x(\tilde{y})$ can be retrieved by solving the associated transportation problem:

\[
\text{(TP) \ Minimize } z(\tilde{y}) = \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \\
\text{subject to } \sum_{j \in J} x_{ij} = d_i, \ i \in I \\
\sum_{i \in I} x_{ij} \leq K_j \tilde{y}_j, \ j \in J \\
x_{ij} \geq 0, \ i \in I, \ j \in J
\]

If $\tilde{y} = y^*$, the optimal location vector, the optimal solution to the original CPLP problem is simply given by $(y^*, x(y^*))$.
Remark 2. An optimal solution of the original CPLP problem can always be found at an extreme point of the polyhedron of feasible flow vectors defined by the constraints:

\[ \sum_{j \in J} x_{ij} = d_i, \ i \in I \]

\[ \sum_{i \in I} x_{ij} \leq K_j, \ j \in J \]

\[ x_{ij} \geq 0, \ i \in I, \ j \in J \]

This property follows from the fact that the CPLP can be interpreted as a fixed-charge problem defined in the space of the flow variables. This fixed-charge problem has a concave objective function that always admits an extreme point minimum. The optimal values for the location variables can easily be obtained from the optimal flow vector by setting \( y_j \) equal to 1 if \( \sum_{i \in I} x_{ij} > 0 \), and to 0 otherwise.
SEARCH SPACES AND NEIGHBOURHOODS FOR THE CPLP

Search space:
1) Full feasible space defined by all variables.
2) Space defined by location variables.
3) Set of extreme points of the set of feasible flow vectors.

Neighbourhoods:

- Depend upon the search space chosen.
- For 2), one can use “Add/Drop” and/or “Swap” neighbourhoods.
- For 3), moves defined by the application of pivots to the linear programming formulation of the transportation problem, since each pivot operation moves the current solution to an adjacent extreme point.
A TEMPLATE FOR LOCAL SEARCH

To maximize $f(S)$ over some domain

Define: $S$, current solution,

$f^*$, value of the best-known solution,

$S^*$, this solution,

$N(S)$, the "neighbourhood" of $S$ (solutions obtained from $S$ by a single transformation).

**Initialization**

*Choose (construct) an initial solution* $S_0$

Set $S := S_0$, $f^* := f(S_0)$, $S^* := S_0$.

**Search**

*While* local optimum not reached *do*

- $S := \arg\max_{S' \in N(S)} [f(S')]$

- *if* $f(S) > f^*$, *then* $f^* := f(S), S^* := S$. 
MAIN CLASSES OF LOCAL SEARCH METHODS

Simple Local Search

• The simplest of all LS approaches
• Consists in constructing a single initial solution and improving it using a single neighbourhood structure until a local optimum is encountered.
• Two variants of simple LS:
  – “Best improvement”
  – “First improvement”

Multi-start Local Search

• A simple extension to the simple LS scheme
• Several (usually randomly generated) initial solutions
• Apply to each of them this simple scheme, thus obtaining several local optima from which the best is selected and returned as the heuristic solution.
SIMULATED ANNEALING

• Kirkpatrick, Gelatt and Vecchi (1983)
• Based on an analogy with the cooling of material in a heat bath.
• Metropolis’ algorithm (1953)
• Solutions $\leftrightarrow$ Configurations of particles
• Objective function $\leftrightarrow$ Energy of system
• Can be interpreted as a controlled random walk in the space of solutions:
  - Improving moves are always accepted;
  - Deteriorating moves are accepted with a probability that depends on the amount of the deterioration and on the temperature (a parameter that decreases with time).
• Extensions/generalizations: deterministic annealing, threshold acceptance methods.
• Local search methods in which deterioration of the objective up to a threshold is accepted.
• As in SA, the threshold decreases as the algorithm progresses.
VARIABLE NEIGHBOURHOOD SEARCH

- Use, instead of a single neighbourhood, several of these in pre-defined sequences.
- Over time VNS has yielded several variants of different complexity.
- The simplest one, called Variable Neighbourhood (VND), is clearly the multi-neighbourhood extension of LS.
- In VND, one first performs LS using the first neighbourhood structure until a local optimum is encountered; the search is then continued using the second neighbourhood structure until a local optimum (w.r.t. to that structure) is encountered, at which point, it switches to the third neighbourhood structure, and so on in a circular fashion.
- VND will eventually stop, but only in a point which is a local optimum for each of the considered neighbourhood structures.
THE TABU SEARCH APPROACH

• Glover (1977, 1986)

• Hansen (1986: steepest ascent/mildest descent)

• A metaheuristic that controls an inner heuristic designed for the specific problem that is to be solved.

• Artificial intelligence concepts: maintain a history of the search in a number of memories.

• Basic principle: allow non-improving moves to overcome local optimal (i.e. keep on transforming the current solution...).

• PROBLEM: How can CYCLING be avoided???

SOLUTION: Keep a HISTORY of the searching process and prohibit «comebacks» to previous solutions (tabu moves).
TABUS

- A short-term memory of the search (in general, only a fixed amount of information is recorded).

- Several possibilities:

  - a list of the last solutions encountered (expensive, and not frequently used);

  - a list of the last modifications performed on current solutions; reverse modifications are then prohibited (the most common type of tabus);

  - a list of key characteristics of the solutions or of the transformations (sometimes more efficient)
EXAMPLES OF TABUS

Consider the situation where one is solving the TSP with 2-opt as inner heuristic.

The basic set of transformations at each step consists of moves obtained by removing two edges \([i, j), (k, \ell)]\; and replacing them with edges \([i, k), (j, \ell)]\.

Possible tabus

- Forbid tours themselves.
- Forbid reverse transformations \([i, k), (j, \ell)] \rightarrow [i, j), (k, \ell)]\ for a few iterations.
- Forbid any transformation involving either \((i, k)\) or \((j, \ell)\) for some time.
- ...
MORE ON TABUS

- **Multiple tabu lists** can be used and have proved quite useful in many contexts.

- “Straightforward” tabus can be implemented as circular lists of fixed length.

- Fixed-length tabus cannot always prevent cycling: many authors have proposed schemes to vary tabu list length during execution (Skorin-Kapov, Taillard).

- Another solution: **random tabu tags**, the duration of a tabu status is a random variable generated when the tabu is created.

- Yet another solution: **randomly activated tabus**, at each iteration, a random number is generated indicating how far to look back in the tabu list (which is otherwise managed like a fixed-length list).
ASPIRATION CRITERIA

● Tabus are sometimes too “powerful”:
  - attractive moves are prohibited, even when there is no danger of cycling;
  - they can lead to overall stagnation of the searching process.

● Aspiration criteria are algorithmic devices that cancel tabus in some circumstances.

● The simplest aspiration criterion consists in allowing a move if it results in a solution with objective value better than that of the best-known solution.

● Much more complicated criteria have been proposed and implemented in some applications.

**KEY RULE**: If cycling cannot occur, you may disregard tabus
SIMPLE TABU SEARCH

To maximize \( f(S) \) over some domain

Define: \( S \), current solution,

\( f^* \), value of the best-known solution,

\( S^* \), this solution,

\( T \), the tabu list,

\( N(S) \), the "neighbourhood" of \( S \) (solutions obtained from \( S \) by a single transformation),

\( \overline{N}(S) \), "admissible" subset of \( N(S) \) (non-tabu or allowed by aspiration).

Initialization

Choose (construct) an initial solution \( s_0 \)

Set \( S := S_0 \), \( f^* := f(S_0) \), \( S^* := S_0 \), \( T := \emptyset \)

Search

While termination criterion not satisfied do

- \( S \in \operatorname{arg\max}_{S' \in \overline{N}(S)} [f(S')] \);
- if \( f(S) > f^* \), then \( f^* := f(S) \), \( S^* := S \);
- record tabu for the current move in \( T \) (delete oldest tabu if necessary).


TERMINATION CRITERIA

• In theory, the search could go on for ever (unless the optimal value of the problem is known beforehand).

• In practice, the search has to be stopped at some point:

  - after a fixed number of iterations (or a fixed amount of CPU time),

  - after some number of iterations without an improvement in the best objective value (probably the most commonly used criterion),

  - when the objective reaches a pre-specified threshold value.

• In complex tabu search schemes, the search will usually be stopped after completing a sequence of **phases**, the duration of each phase being determined by one of the above criteria.
PROBABILISTIC TABU SEARCH

In “regular” simple tabu search, one must evaluate the objective for every element in the neighbourhood $N(S)$ of the current solution.

Instead of considering the whole set $N(S)$, one may restrict its attention to a random sample $N'(S) \subset N(S)$.

**Advantages:**

- In most applications, a smaller computational effort, since one only evaluates the objective for $S' \in N'(S)$;

- The random choice of $N'(S)$ acts as an anti-cycling choice $\rightarrow$ shorter tabu lists can be used.

**Disadvantage:** the best solution may be missed.
SEARCH INTENSIFICATION

Idea: To explore more thoroughly portions of the search space that seem “promising”

- From times to times, the normal searching process is stopped and an intensification phase is executed.

- Often based on some kind of intermediate-term memory → recency memory records the number of iterations that “elements” have been present in the current solution.

- Often restarted from the best-known solution.

- Possible techniques:
  - “freezing” (fixing) “good” elements in the current solution;
  - changing (increasing) sample size in probabilistic TS;
  - switching to a different inner heuristic or modifying the parameters driving it.
SEARCH DIVERSIFICATION

- In many cases, the normal searching process tends to spend most of its time in a restricted portion of the search space. Good solutions may be obtained, but one may still be far from the optimum.

  *Diversification*: a mechanism to “force” the search into previously unexplored areas.

- Usually based on some form of **long-term memory**. 
  → **frequency memory** records the number of times each “element” has appeared in the solution.

- Most common techniques:

  - **restart diversification**: force a few “unfrequent” elements in the solution and restart the search from the new current solution thus obtained;

  - **continuous diversification**: in the evaluation of moves, bias the objective by adding a small term related to element frequencies;

  - strategic oscillation: (see next transparency).
HANDLING CONSTRAINTS

● In many instances, accounting for all problem constraints in the definition of the search space severely restricts the search process and leads to mediocre solutions.

→ *constraint relaxation is often effective!*

● “Wider” search space which is often easier to handle
→ simpler neighbourhoods can be used.

● Constraint violations are added to the objective as a weighted penalty term.

● But, how can one find “good” weights?

→ **self-adjusting penalties** can be used

- weights are adjusted dynamically based on the recent history of the search
  + increase weights when only infeasible solutions are encountered,
  + decrease weights if the opposite occurs.

**Strategic oscillation**: changing weights to induce diversification.
SURROGATE AND AUXILIARY OBJECTIVES

• In some problems, the true objective function is extremely costly to evaluate (e.g., MIP, with the search space restricted to integer variables; stochastic programming;...).

  → The evaluation of moves becomes prohibitive (even if sampling is used).

• Solution: evaluate neighbours using a surrogate objective function

  - correlated to the true objective,

  - less demanding computationally,

  - the value of the true objective is computed only for the chosen move or for a subset of promising candidates.

• In some problems, most neighbours have the same objective value. How can one choose the next move among them?

  By using an auxiliary objective function measuring a desirable attribute of solutions.
RECENT TRENDS IN TABU SEARCH (AND OTHER LOCAL SEARCH APPROACHES)
PARALLEL VARIANTS

Parallel processing opens up great opportunities for new developments in tabu search.

- **Low-level parallelization**
  Using parallel processing to speed up computationally demanding steps of “standard” tabu search.

- **High-level parallelization**
  Run several search threads in parallel to obtain more information and come up with better solutions (parallel search threads can also be used on sequential architectures).

These techniques have already been used with very good results.


HYBRIDS

Using local or tabu search in combination with other optimization techniques.

- In branch-and-bound, to compute bounds.
- In conjunction with genetic algorithms or ant colony optimization.
- Alternately with other LS or TS methods.
- In conjunction with Constraint Logic Programming techniques.

Currently, the most successful methods.

Two general schemes:

- “unified” architectures (a single algorithm combining components of several methods),
- “parallel hybrids” (running concurrently “pure” implementations of two or more algorithms).
USING INFORMATION IN A DIFFERENT WAY

- **Reactive Tabu Search**  
  - Battiti and Tecchiolli (1992, 1994)

- **Path relinking, Scatter search**  
  - Glover (1994, 1995)  
  - Glover and Laguna (1997)

- **Candidate list and elite solutions**  
  - see Glover and Laguna (1997)

- **Hashing and Chunking**  
  - Woodruff and Zemel (1993)  
  - Carlton and Barnes (1995)  
  - Woodruff (1996)

- **Vocabulary building**  
  - Glover (1992)  
  - Glover and Laguna (1993)  
  - Rochat and Taillard (1995)  
  - Kelly and Xu (1995)  
NEW APPLICATION AREAS

• Integer and mixed-integer programming

• Continuous optimization problems
  - with extreme point solutions
    + concave programming
    + fixed-charge problems
  - with “general” solution structure

• Continuous, multi-criteria optimization

• Stochastic programming problems
  especially those with a large number of possible realizations (intractable using standard approaches)

• Real-time decision problems
  - LS methods almost possess the “Anytime” property;
  - Solutions can often be adjusted in real time to new information.
IN-DEPTH PERFORMANCE ANALYSIS

• New area launched about 5 years ago by Jean-Paul Watson and his co-authors.

• The focus is not on developing new methods, but in modelling and understanding the behaviour of existing methods.
LOCAL SEARCH OPERATORS
IN ROUTING
**2-opt exchange operator**

Edges \((i, i+1)\) and \((j, j+1)\) are replaced by edges \((i, j)\) and \((i+1, j+1)\), thus reversing the direction of customers between \(i+1\) and \(j\).
**Or-opt operator**

Customers $i$ and $i+1$ are relocated to be served between two customers $j$ and $j+1$ instead of customers $i-1$ and $i+2$. This is performed by replacing 3 edges $(i-1, i)$, $(i+1, i+2)$ and $(j, j+1)$ by the edges $(i-1, i+2)$, $(j, i)$ and $(i+1, j+1)$, preserving the orientation of the route.
2-opt* operator

The customers served after customer $i$ on the upper route are reinserted to be served after customer $j$ on the lower route and customers after $j$ on the lower route are moved to be served on the upper route after customer $i$. This is performed by replacing edges $(i, i+1)$ and $(j, j+1)$ with edges $(i, j+1)$ and $(j, i+1)$. 
Relocate operator

Edges \((i-1, i), (i, i+1)\) and \((j, j+1)\) are replaced by \((i-1, i+1)\), 
\((j, i)\) and \((i, j+1)\), i.e., customer \(i\) from the origin route is placed into the destination route.
**Exchange operator**

Edges \((i-1, i), (i, i+1), (j-1, j)\) and \((j, j+1)\) are replaced by \((i-1, j), (j, i+1), (j-1, i)\) and \((i, j+1)\), i.e., two customers from different routes are simultaneously placed into the other routes.
**CROSS-exchange**

Segments \((i, k)\) on the upper route and \((j, l)\) on the lower route are simultaneously reinserted into the lower and upper routes, respectively. This is performed by replacing edges \((i-1, i), (k, k+1), (j-1, j)\) and \((l, l+1)\) by edges \((i-1, j), (l, k+1), (j-1, i)\) and \((k, l+1)\). Note that the orientation of both routes is preserved.
**GENI-exchange operator**

Customer $i$ on the upper route is inserted into the lower route between the customers $j$ and $k$ closest to it by adding the edges $(j, i)$ and $(i, k)$. Since $j$ and $k$ are not consecutive, one has to reorder the lower route. Here the feasible tour is obtained by deleting edges $(j, j+1)$ and $(k-1, k)$ and by relocating the path \{j+1,…, k-1\}. 
**Cyclic transfer operator**

The basic idea is to transfer simultaneously the customers denoted by white circles in cyclical manner between the routes. More precisely here customers $a$ and $c$ in route 1, $f$ and $j$ in route 2 and $o$ and $p$ in route 4 are simultaneously transferred to routes 2, 4, and 1 respectively, and route 3 remains untouched.
LOCAL SEARCH METHODS FOR THE VRP
EARLY METHODS

SIMULATED ANNEALING

• Robusté et al. (1990):
  − Complex neighbourhood (swap + Or-opt + …)
  − Only tested on four instances

• Alfa et al. (1991):
  − Route-first, cluster second heuristic for the initial solution
  − 3-opt neighbourhood
  − Not competitive

• Osman (1993):
  − $\lambda$-interchange neighbourhood (includes swaps and relocate for subsets of size $\leq \lambda$)
  − Special cooling schedule
  − Generally produces good, but not exceptional results

• Van Breedam (1995):
  − Tested several variants of SA
  − Could not match results produced with Tabu Search
EARLY METHODS (2)

TABU SEARCH

• Willard (1989):
  – Solution represented as a giant tour by replication of the depot
  – Neighbourhood based on 2- and 3-opt moves
  – Does not seem competitive

• Pureza and França (1991):
  – (Relocate + swap) neighbourhood
  – Preserve feasibility
  – Did not produce exceptionally good results
MEDIEVAL METHODS

TABU SEARCH

- Osman (1993):
  - $\lambda$-interchange neighbourhood with $\lambda = 2$
  - Variants with “best accept” and “first improvement” rules
  - Generally produces excellent, but not the best results

- Taillard (1993):
  - $\lambda$-interchange neighbourhood
  - Feasible solutions
  - Decomposition into smaller subproblems that are modified during the execution of the algorithm
  - Suitable for parallel implementations
  - Continuous diversification
  - Excellent computational results, but unknown CPU times
MEDIEVAL METHODS (2)

TABU SEARCH

- TABUROUTE (Gendreau, Hertz, Laporte 1994)
  - GENI neighbourhoods
  - Moves in infeasible space with self-adjusting penalties
  - Continuous diversification
  - Random tabu tags
  - Excellent computational results

- Rochat and Taillard (1995):
  - Introduces the concept of adaptive memory (pool of elite solutions used to reconstruct solutions for intensification/diversification purposes)
  - Outstanding computational results on both the VRP and the VRPTW

- Rego and Roucairol (1996)
  - Based on ejection chains (cyclic transfer neighbourhood)
  - Parallel implementation
  - Generally produces excellent, but not the best results
MEDIEVAL METHODS (3)

DETERMINISTIC ANNEALING

- Golden et al. (1998):
  - Applied record-to-record travel to 20 large instances
  - Produces better results than Xu and Kelly’s tabu search heuristic for 11 instances out of 20
  - Much faster than Xu and Kelly’s heuristic
RECENT METHODS

**TABU SEARCH**

- **Granular Tabu Search** (Toth and Vigo 2003)
  - Removes from the graph long edges unlikely to belong to the optimal solution
  - Typically keep between 10 to 20% of the original edges
  - The *sparsification parameter* can be adjusted dynamically to yield intensification or diversification
  - Edge-exchange neighbourhood
  - Excellent results (see tables later)

  - Similar in many ways to TABUROUTE
  - A single initial solution is considered
  - Additional diversification is used by moving the depot arbitrarily at some points
  - Can be applied to many variants of the VRP
  - Excellent computational results

- **Bone Route** (Tarantilis and Kiranoudis 2002)
  - Tabu search procedure with an adaptive memory
  - 2-opt, swap and relocate moves
  - New solutions are constructed from the adaptive memory by extracting route segments, called *bones*, from high quality routes.
  - Excellent results
RECENT METHODS

DETERMINISTIC ANNEALING

• Li et al. (2004):
  – Combines record-to-record principles with a variable-length neighbour list whose principle is similar to Granular Tabu Search
  – Neighbourhood based on intra-route and inter-route 2-opt moves
  – Excellent results

VERY LARGE NEIGHBORHOOD SEARCH

• Ergun et al. (2003):
  – Descent mechanism
  – The method considers at each iteration a composite neighbourhood involving changes to several routes as in ejection chains or the cyclic transfer neighbourhood
  – Changes to individual routes are based on 2-opt, swap and relocate moves.
  – The set of moves to be performed at each iteration is obtained by solving a shortest path problem.
  – Excellent results
Table 1.1. Computational results for the Christofides et al. (1979) instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>n Type</th>
<th>Value</th>
<th>%</th>
<th>Minutes</th>
<th>Value</th>
<th>%</th>
<th>Minutes</th>
<th>Value</th>
<th>%</th>
<th>Minutes</th>
<th>Value</th>
<th>%</th>
<th>Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50 C</td>
<td>524.61</td>
<td>0.00</td>
<td>0.81</td>
<td>524.61</td>
<td>0.00</td>
<td>2.32</td>
<td>524.61</td>
<td>0.00</td>
<td>23.13</td>
<td>524.61</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>2</td>
<td>75 C</td>
<td>838.60</td>
<td>0.40</td>
<td>2.21</td>
<td>836.18</td>
<td>0.11</td>
<td>14.78</td>
<td>835.28</td>
<td>0.00</td>
<td>26.66</td>
<td>835.43</td>
<td>0.02</td>
<td>33.93</td>
</tr>
<tr>
<td>3</td>
<td>100 C</td>
<td>828.56</td>
<td>0.29</td>
<td>2.39</td>
<td>827.39</td>
<td>0.15</td>
<td>11.67</td>
<td>827.46</td>
<td>0.16</td>
<td>21.30</td>
<td>826.14</td>
<td>0.00</td>
<td>0.46</td>
</tr>
<tr>
<td>4</td>
<td>150 C</td>
<td>1033.21</td>
<td>0.47</td>
<td>4.51</td>
<td>1045.36</td>
<td>1.65</td>
<td>26.66</td>
<td>1032.68</td>
<td>0.41</td>
<td>24.45</td>
<td>1031.63</td>
<td>0.31</td>
<td>5.50</td>
</tr>
<tr>
<td>5</td>
<td>199 C</td>
<td>1318.25</td>
<td>2.09</td>
<td>7.50</td>
<td>1303.47</td>
<td>0.94</td>
<td>57.68</td>
<td>1307.33</td>
<td>1.24</td>
<td>57.25</td>
<td>1300.23</td>
<td>0.69</td>
<td>19.10</td>
</tr>
<tr>
<td>6</td>
<td>50 C, D</td>
<td>555.43</td>
<td>0.00</td>
<td>0.86</td>
<td>555.43</td>
<td>0.00</td>
<td>3.03</td>
<td>555.43</td>
<td>0.00</td>
<td>3.50</td>
<td>555.43</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>7</td>
<td>75 C, D</td>
<td>920.72</td>
<td>1.21</td>
<td>2.75</td>
<td>909.68</td>
<td>0.00</td>
<td>7.41</td>
<td>910.04</td>
<td>0.04</td>
<td>36.53</td>
<td>912.30</td>
<td>0.29</td>
<td>1.42</td>
</tr>
<tr>
<td>8</td>
<td>100 C, D</td>
<td>869.48</td>
<td>0.41</td>
<td>2.90</td>
<td>865.95</td>
<td>0.00</td>
<td>10.93</td>
<td>865.94</td>
<td>0.00</td>
<td>12.43</td>
<td>865.94</td>
<td>0.00</td>
<td>0.37</td>
</tr>
<tr>
<td>9</td>
<td>150 C, D</td>
<td>1173.12</td>
<td>0.91</td>
<td>5.67</td>
<td>1167.85</td>
<td>0.46</td>
<td>51.66</td>
<td>1164.88</td>
<td>0.20</td>
<td>42.47</td>
<td>1164.25</td>
<td>0.15</td>
<td>7.25</td>
</tr>
<tr>
<td>10</td>
<td>199 C, D</td>
<td>1435.74</td>
<td>2.86</td>
<td>9.11</td>
<td>1416.84</td>
<td>1.50</td>
<td>106.28</td>
<td>1404.36</td>
<td>0.61</td>
<td>28.32</td>
<td>1402.20</td>
<td>1.74</td>
<td>26.83</td>
</tr>
<tr>
<td>11</td>
<td>120 C</td>
<td>1042.87</td>
<td>0.07</td>
<td>3.18</td>
<td>1042.11</td>
<td>0.00</td>
<td>11.67</td>
<td>1042.11</td>
<td>0.00</td>
<td>69.13</td>
<td>1042.11</td>
<td>0.00</td>
<td>0.30</td>
</tr>
<tr>
<td>12</td>
<td>100 C</td>
<td>819.56</td>
<td>0.00</td>
<td>1.10</td>
<td>819.56</td>
<td>0.00</td>
<td>9.02</td>
<td>819.56</td>
<td>0.00</td>
<td>5.98</td>
<td>819.56</td>
<td>0.00</td>
<td>0.05</td>
</tr>
<tr>
<td>13</td>
<td>120 C, D</td>
<td>1545.51</td>
<td>0.28</td>
<td>9.34</td>
<td>1549.25</td>
<td>0.53</td>
<td>21.00</td>
<td>1544.99</td>
<td>0.25</td>
<td>39.73</td>
<td>1542.97</td>
<td>0.12</td>
<td>10.44</td>
</tr>
<tr>
<td>14</td>
<td>100 C, D</td>
<td>866.37</td>
<td>0.00</td>
<td>1.41</td>
<td>866.37</td>
<td>0.00</td>
<td>10.53</td>
<td>866.37</td>
<td>0.00</td>
<td>6.55</td>
<td>866.37</td>
<td>0.00</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Average 0.64 3.84 0.41 0.56 24.62 0.23 28.91 0.24 5.19

1. C: Capacity restrictions; D: Route length restrictions.
2. Pentium (200 MHz).
3. Best variant ($\alpha = 0.4$).
4. Results of recent computational experiments (see Section 3.3); the average % deviation in Cordeau et al. (2001) is 0.69.
5. Pentium IV (2GHz).
6. Best of five runs.
7. Time for reaching the best value for the first time (Pentium III, 733 MHz).
8. GHz PC (75 MFlops).
Table 1.1. Computational results for the Christofides et al. (1979) instances

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Value</td>
<td>%</td>
<td>Minutes</td>
<td>Value</td>
<td>%</td>
</tr>
<tr>
<td>1</td>
<td>50</td>
<td>C</td>
<td>524.61 0.00</td>
<td>0.11</td>
<td>524.61 0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>2</td>
<td>75</td>
<td>C</td>
<td>835.26 0.00</td>
<td>4.56</td>
<td>835.26 0.00</td>
<td>0.26</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>C</td>
<td>826.14 0.00</td>
<td>7.66</td>
<td>826.14 0.00</td>
<td>0.05</td>
</tr>
<tr>
<td>4</td>
<td>150</td>
<td>C</td>
<td>1030.88 0.24</td>
<td>9.13</td>
<td>1028.42 0.00</td>
<td>0.47</td>
</tr>
<tr>
<td>5</td>
<td>199</td>
<td>C</td>
<td>1314.11 1.77</td>
<td>16.97</td>
<td>1291.29 0.00</td>
<td>101.93</td>
</tr>
<tr>
<td>6</td>
<td>50</td>
<td>C, D</td>
<td>555.43 0.00</td>
<td>0.10</td>
<td>555.43 0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>7</td>
<td>75</td>
<td>C, D</td>
<td>909.68 0.00</td>
<td>0.92</td>
<td>909.68 0.00</td>
<td>0.43</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>C</td>
<td>865.94 0.00</td>
<td>4.28</td>
<td>865.94 0.00</td>
<td>0.44</td>
</tr>
<tr>
<td>9</td>
<td>150</td>
<td>C, D</td>
<td>1163.19 0.06</td>
<td>5.83</td>
<td>1162.55 0.00</td>
<td>1.22</td>
</tr>
<tr>
<td>10</td>
<td>199</td>
<td>C, D</td>
<td>1408.82 0.93</td>
<td>14.32</td>
<td>1401.12 0.41</td>
<td>2.45</td>
</tr>
<tr>
<td>11</td>
<td>120</td>
<td>C</td>
<td>1042.11 0.00</td>
<td>0.21</td>
<td>1042.11 0.00</td>
<td>0.05</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>C</td>
<td>819.56 0.00</td>
<td>0.10</td>
<td>819.56 0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>13</td>
<td>120</td>
<td>C, D</td>
<td>1544.01 0.19</td>
<td>8.75</td>
<td>1541.14 0.00</td>
<td>0.63</td>
</tr>
<tr>
<td>14</td>
<td>100</td>
<td>C, D</td>
<td>866.37 0.00</td>
<td>0.10</td>
<td>866.37 0.00</td>
<td>0.08</td>
</tr>
</tbody>
</table>

Average: 0.23 | 5.22 | 0.03 | 7.72 | 0.07 | 0.27 | 0.49 | 21.25

9. Pentium II (400 MHz).
10. For C instances, see Mester and Bräysy (2004). Otherwise, see Mester (2004).
11. Pentium IV (2 GHz).
12. Pentium (400 MHz).
<table>
<thead>
<tr>
<th>Instance</th>
<th>Type</th>
<th>Value</th>
<th>Minutes</th>
<th>Value</th>
<th>Minutes</th>
<th>Value</th>
<th>Minutes</th>
<th>Value</th>
<th>Minutes</th>
<th>Value</th>
<th>Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 240 C</td>
<td></td>
<td>5736.15</td>
<td>1.93</td>
<td>4.98</td>
<td>5666.42</td>
<td>0.69</td>
<td>5681.97</td>
<td>0.97</td>
<td>10.29</td>
<td>5741.79</td>
<td>2.03</td>
</tr>
<tr>
<td>2 320 C</td>
<td></td>
<td>8553.01</td>
<td>1.24</td>
<td>8.28</td>
<td>8469.32</td>
<td>0.25</td>
<td>8657.36</td>
<td>2.48</td>
<td>35.39</td>
<td>8917.41</td>
<td>5.56</td>
</tr>
<tr>
<td>3 400 C</td>
<td></td>
<td>11402.75</td>
<td>3.32</td>
<td>12.94</td>
<td>11145.80</td>
<td>0.99</td>
<td>11037.40</td>
<td>0.01</td>
<td>55.39</td>
<td>12106.64</td>
<td>9.70</td>
</tr>
<tr>
<td>4 480 C</td>
<td></td>
<td>14910.62</td>
<td>9.44</td>
<td>15.13</td>
<td>13758.08</td>
<td>0.98</td>
<td>13740.60</td>
<td>0.85</td>
<td>83.19</td>
<td>15316.69</td>
<td>12.42</td>
</tr>
<tr>
<td>5 200 C</td>
<td></td>
<td>6697.53</td>
<td>3.66</td>
<td>2.38</td>
<td>6478.09</td>
<td>0.26</td>
<td>6756.44</td>
<td>4.57</td>
<td>5.13</td>
<td>6570.28</td>
<td>1.69</td>
</tr>
<tr>
<td>6 280 C</td>
<td></td>
<td>8963.32</td>
<td>6.54</td>
<td>4.65</td>
<td>8539.61</td>
<td>1.51</td>
<td>8537.17</td>
<td>1.48</td>
<td>18.64</td>
<td>8836.25</td>
<td>5.03</td>
</tr>
<tr>
<td>7 360 C</td>
<td></td>
<td>10547.44</td>
<td>3.45</td>
<td>11.06</td>
<td>10289.72</td>
<td>0.92</td>
<td>10267.40</td>
<td>0.70</td>
<td>25.60</td>
<td>11116.68</td>
<td>9.03</td>
</tr>
<tr>
<td>8 440 C</td>
<td></td>
<td>12036.24</td>
<td>3.20</td>
<td>11.08</td>
<td>11920.52</td>
<td>2.20</td>
<td>11869.50</td>
<td>1.77</td>
<td>71.44</td>
<td>12634.17</td>
<td>8.32</td>
</tr>
<tr>
<td>9 255 C</td>
<td></td>
<td>593.35</td>
<td>1.71</td>
<td>11.67</td>
<td>588.25</td>
<td>0.85</td>
<td>587.39</td>
<td>0.69</td>
<td>37.26</td>
<td>587.89</td>
<td>0.77</td>
</tr>
<tr>
<td>10 323 C</td>
<td></td>
<td>751.66</td>
<td>1.30</td>
<td>15.83</td>
<td>749.49</td>
<td>1.01</td>
<td>752.76</td>
<td>1.45</td>
<td>51.11</td>
<td>749.85</td>
<td>1.05</td>
</tr>
<tr>
<td>11 399 C</td>
<td></td>
<td>936.04</td>
<td>1.92</td>
<td>33.12</td>
<td>925.91</td>
<td>0.81</td>
<td>929.07</td>
<td>1.16</td>
<td>41.54</td>
<td>932.74</td>
<td>1.56</td>
</tr>
<tr>
<td>12 483 C</td>
<td></td>
<td>1147.14</td>
<td>3.61</td>
<td>42.90</td>
<td>1128.03</td>
<td>1.88</td>
<td>1119.52</td>
<td>1.11</td>
<td>157.01</td>
<td>1134.63</td>
<td>2.48</td>
</tr>
<tr>
<td>13 252 C</td>
<td></td>
<td>868.80</td>
<td>1.13</td>
<td>11.43</td>
<td>865.20</td>
<td>0.71</td>
<td>875.88</td>
<td>1.95</td>
<td>34.83</td>
<td>870.90</td>
<td>1.37</td>
</tr>
<tr>
<td>14 320 C</td>
<td></td>
<td>1096.18</td>
<td>1.38</td>
<td>14.51</td>
<td>1097.78</td>
<td>1.52</td>
<td>1102.03</td>
<td>1.92</td>
<td>21.56</td>
<td>1097.11</td>
<td>1.46</td>
</tr>
<tr>
<td>15 396 C</td>
<td></td>
<td>1369.44</td>
<td>1.80</td>
<td>18.45</td>
<td>1361.41</td>
<td>1.20</td>
<td>1363.76</td>
<td>1.38</td>
<td>57.64</td>
<td>1367.15</td>
<td>1.63</td>
</tr>
<tr>
<td>16 480 C</td>
<td></td>
<td>1652.32</td>
<td>1.83</td>
<td>23.07</td>
<td>1635.58</td>
<td>0.79</td>
<td>1647.06</td>
<td>1.50</td>
<td>129.50</td>
<td>1643.00</td>
<td>1.25</td>
</tr>
<tr>
<td>17 240 C</td>
<td></td>
<td>711.07</td>
<td>0.46</td>
<td>14.29</td>
<td>711.74</td>
<td>0.56</td>
<td>710.93</td>
<td>0.44</td>
<td>18.03</td>
<td>716.46</td>
<td>1.22</td>
</tr>
<tr>
<td>18 300 C</td>
<td></td>
<td>1016.83</td>
<td>1.81</td>
<td>21.45</td>
<td>1010.32</td>
<td>1.16</td>
<td>1014.62</td>
<td>1.59</td>
<td>67.11</td>
<td>1023.32</td>
<td>2.46</td>
</tr>
<tr>
<td>19 360 C</td>
<td></td>
<td>1400.96</td>
<td>2.49</td>
<td>30.06</td>
<td>1382.59</td>
<td>1.15</td>
<td>1383.79</td>
<td>1.24</td>
<td>66.21</td>
<td>1404.84</td>
<td>2.78</td>
</tr>
<tr>
<td>20 420 C</td>
<td></td>
<td>1915.83</td>
<td>5.20</td>
<td>43.03</td>
<td>1850.92</td>
<td>1.63</td>
<td>1854.24</td>
<td>1.82</td>
<td>135.29</td>
<td>1883.33</td>
<td>3.41</td>
</tr>
</tbody>
</table>

Average: 2.87 17.55 1.05 1.45 56.11 3.76 137.95 0.91 66.90

1. C: Capacity restrictions; D: Route length restrictions.
2. Pentium (200 MHz).
3. Best variant (α = 0.01).
4. Results of recent computational experiments (see Section 3.3).
5. Pentium IV (2GHz).
6. Best of two runs.
7. Time for reaching the best value for the first time (Pentium III, 733 MHz).
8. GHz PC (75 MFlops).
<table>
<thead>
<tr>
<th>Instance</th>
<th>Type</th>
<th>Value</th>
<th>%</th>
<th>Minutes</th>
<th>Value**</th>
<th>%</th>
<th>Minutes**</th>
<th>Value**</th>
<th>%</th>
<th>Minutes**</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 240 C</td>
<td></td>
<td>5676.97</td>
<td>0.88</td>
<td>27.86</td>
<td><strong>5627.54</strong></td>
<td>8.73</td>
<td>5644.00</td>
<td>0.30</td>
<td>0.70</td>
<td>5644.02</td>
<td>0.29</td>
</tr>
<tr>
<td>2 320 C</td>
<td></td>
<td>8512.64</td>
<td>0.77</td>
<td>55.62</td>
<td><strong>8447.92</strong></td>
<td>46.66</td>
<td>8468.00</td>
<td>0.24</td>
<td>0.20</td>
<td>8449.12</td>
<td>0.01</td>
</tr>
<tr>
<td>3 400 C</td>
<td></td>
<td>11199.72</td>
<td>1.48</td>
<td>59.21</td>
<td><strong>11036.22</strong></td>
<td>40.55</td>
<td>11146.00</td>
<td>0.99</td>
<td>0.70</td>
<td>11036.22</td>
<td>0.00</td>
</tr>
<tr>
<td>4 480 C</td>
<td></td>
<td>13637.53</td>
<td>0.10</td>
<td>47.63</td>
<td><strong>13024.52</strong></td>
<td>470.00</td>
<td>13704.52</td>
<td>0.50</td>
<td>2.50</td>
<td>13099.11</td>
<td>0.55</td>
</tr>
<tr>
<td>5 200 C</td>
<td></td>
<td>5676.97</td>
<td>0.88</td>
<td>11.34</td>
<td><strong>6460.98</strong></td>
<td>0.17</td>
<td>6466.00</td>
<td>0.08</td>
<td>0.50</td>
<td>6460.98</td>
<td>0.00</td>
</tr>
<tr>
<td>6 280 C</td>
<td></td>
<td>8429.28</td>
<td>0.20</td>
<td>12.54</td>
<td><strong>8412.88</strong></td>
<td>75.22</td>
<td>8539.61</td>
<td>1.51</td>
<td>0.10</td>
<td>8412.90</td>
<td>0.00</td>
</tr>
<tr>
<td>7 360 C</td>
<td></td>
<td>10216.50</td>
<td>0.21</td>
<td>42.50</td>
<td><strong>10195.56</strong></td>
<td>2.55</td>
<td>10240.42</td>
<td>0.44</td>
<td>0.85</td>
<td>10195.59</td>
<td>0.00</td>
</tr>
<tr>
<td>8 440 C</td>
<td></td>
<td>11936.16</td>
<td>2.34</td>
<td>79.69</td>
<td><strong>11663.55</strong></td>
<td>34.30</td>
<td>11918.75</td>
<td>2.19</td>
<td>0.27</td>
<td>11828.78</td>
<td>1.42</td>
</tr>
<tr>
<td>9 255 C,D</td>
<td></td>
<td>583.39</td>
<td>0.00</td>
<td>8.33</td>
<td>588.25</td>
<td>0.83</td>
<td>586.87</td>
<td>0.60</td>
<td>21.52</td>
<td>583.39</td>
<td></td>
</tr>
<tr>
<td>10 323 C,D</td>
<td></td>
<td>742.03</td>
<td>0.00</td>
<td>6.00</td>
<td>752.92</td>
<td>1.39</td>
<td>750.77</td>
<td>1.25</td>
<td>17.48</td>
<td>742.03</td>
<td></td>
</tr>
<tr>
<td>11 399 C,D</td>
<td></td>
<td>918.45</td>
<td>0.00</td>
<td>110.00</td>
<td>925.94</td>
<td>0.82</td>
<td>927.27</td>
<td>0.96</td>
<td>96.88</td>
<td>918.45</td>
<td></td>
</tr>
<tr>
<td>12 483 C,D</td>
<td></td>
<td>1107.19</td>
<td>0.00</td>
<td>600.00</td>
<td>1126.67</td>
<td>1.94</td>
<td>1140.87</td>
<td>3.04</td>
<td>61.38</td>
<td>1107.19</td>
<td></td>
</tr>
<tr>
<td>13 252 C,D</td>
<td></td>
<td>859.11</td>
<td>0.00</td>
<td>10.25</td>
<td>865.20</td>
<td>0.71</td>
<td>865.07</td>
<td>0.69</td>
<td>87.20</td>
<td>859.11</td>
<td></td>
</tr>
<tr>
<td>14 320 C,D</td>
<td></td>
<td>1081.31</td>
<td>0.00</td>
<td>1.22</td>
<td>1097.68</td>
<td>1.51</td>
<td>1093.77</td>
<td>1.15</td>
<td>25.85</td>
<td>1081.31</td>
<td></td>
</tr>
<tr>
<td>15 396 C,D</td>
<td></td>
<td>1345.23</td>
<td>0.00</td>
<td>7.17</td>
<td>1354.76</td>
<td>0.71</td>
<td>1358.21</td>
<td>0.96</td>
<td>23.80</td>
<td>1345.23</td>
<td></td>
</tr>
<tr>
<td>16 480 C,D</td>
<td></td>
<td>1622.69</td>
<td>0.00</td>
<td>20.00</td>
<td>1634.99</td>
<td>0.76</td>
<td>1635.16</td>
<td>0.77</td>
<td>39.90</td>
<td>1622.69</td>
<td></td>
</tr>
<tr>
<td>17 240 C,D</td>
<td></td>
<td>707.79</td>
<td>0.00</td>
<td>0.75</td>
<td>710.22</td>
<td>0.34</td>
<td>708.76</td>
<td>0.14</td>
<td>68.50</td>
<td>707.79</td>
<td></td>
</tr>
<tr>
<td>18 300 C,D</td>
<td></td>
<td>998.73</td>
<td>0.00</td>
<td>2.50</td>
<td>1009.53</td>
<td>1.08</td>
<td>998.83</td>
<td>0.01</td>
<td>42.73</td>
<td>998.73</td>
<td></td>
</tr>
<tr>
<td>19 360 C,D</td>
<td></td>
<td>1366.86</td>
<td>0.00</td>
<td>6.00</td>
<td>1381.88</td>
<td>1.10</td>
<td>1367.20</td>
<td>0.02</td>
<td>112.80</td>
<td>1366.86</td>
<td></td>
</tr>
<tr>
<td>20 420 C,D</td>
<td></td>
<td>1821.15</td>
<td>0.00</td>
<td>8.40</td>
<td>1840.57</td>
<td>1.03</td>
<td>1822.94</td>
<td>0.10</td>
<td>71.42</td>
<td>1821.15</td>
<td></td>
</tr>
</tbody>
</table>

Average: 0.74 42.05 0.00 72.94 0.93 0.63 0.60 49.33

9. Pentium II (400 MHz).
10. For C instances, see Mester and Bräysy (2004). Otherwise, see Mester (2004).
11. Pentium IV (2GHz).
12. Best value obtained in several experiments.
13. Pentium (900 MHz).
LOCAL SEARCH METHODS FOR THE VRPTW
TABU SEARCH FOR VRPTW

• Initial solution: typically created with some cheapest insertion heuristic.

• Improvement using local search with one or more neighborhood structures and the best-accept strategy. Most of the neighborhoods used are well known.

• To reduce the complexity of the search, some authors propose special strategies for limiting the neighborhood.

• To cross the barriers of the search space, created by time window constraints, some authors allow infeasibilities during the search. The violations of constraints are penalized in the cost function and the parameter values regarding each type of violation are adjusted dynamically.

• Since the number of routes is often considered as the primary objective, some authors use different explicit strategies for reducing the number of routes.

• Most of the proposed tabu searches use specialized diversification and intensification strategies to guide the search (e.g., “adaptive memory”, Rochat and Taillard, 1995).

• Several authors report using various post-optimization techniques.
## THE MAIN FEATURES OF TABU SEARCH HEURISTICS FOR VRPTW

<table>
<thead>
<tr>
<th>Authors</th>
<th>Year</th>
<th>Initial solution</th>
<th>Neighborhood Operators</th>
<th>Route min.</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Garcia et al.</td>
<td>1994</td>
<td>Solomon’s I1 heuristic</td>
<td>2-opt*, Or-opt</td>
<td>Yes</td>
<td>Neighborhood restricted to arcs close in distance</td>
</tr>
<tr>
<td>Rochat et al.</td>
<td>1995</td>
<td>Modification of Solomon’s I1, 2-opt relocate</td>
<td>No</td>
<td>Reactive tabu search</td>
<td></td>
</tr>
<tr>
<td>Carlton</td>
<td>1995</td>
<td>Insertion heuristic</td>
<td>relocate</td>
<td>No</td>
<td>Neighborhood restricted to arcs close in distance</td>
</tr>
<tr>
<td>Potvin et al.</td>
<td>1996</td>
<td>Solomon’s I1 heuristic</td>
<td>2-opt*, Or-opt</td>
<td>Yes</td>
<td>Soft time windows, adaptive memory</td>
</tr>
<tr>
<td>Taillard et al.</td>
<td>1997</td>
<td>Solomon’s I1 heuristic</td>
<td>CROSS</td>
<td>No</td>
<td>Soft time windows, adaptive memory, Reactive tabu search</td>
</tr>
<tr>
<td>Badeau et al.</td>
<td>1997</td>
<td>Solomon’s I1 heuristic</td>
<td>CROSS</td>
<td>No</td>
<td>Reactive tabu search</td>
</tr>
<tr>
<td>De Backer et</td>
<td>1997</td>
<td>Savings heuristic</td>
<td>exchange, relocate,</td>
<td>No</td>
<td>Constraint programming</td>
</tr>
<tr>
<td>Authors</td>
<td>Year</td>
<td>Heuristic Type</td>
<td>Moves</td>
<td>Constraint Based Diversification</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
<td>-----------------------------------------</td>
<td>--------------------------------------------</td>
<td>----------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Brandão et al.</td>
<td>1999</td>
<td>Insertion heuristic</td>
<td>relocate, exchange, GENI</td>
<td>No</td>
<td>used to check feasibility of moves</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Neighborhoods restricted to arcs close in distance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Generated routes stored in a pool</td>
</tr>
<tr>
<td>Schulze et al.</td>
<td>1999</td>
<td>Solomon’s I1, parallel I1 and savings heuristic</td>
<td>Ejection chains, Or-opt</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Tan et al.</td>
<td>2000</td>
<td>Insertion heuristic of Thangiah (1994)</td>
<td>$\lambda$-interchange, 2-opt*</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Lau et al.</td>
<td>2000</td>
<td>Insertion heuristic</td>
<td>exchange, relocate</td>
<td>No</td>
<td>Constraint based diversification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cordeau et al.</td>
<td>2001</td>
<td>Modification of Sweep heuristic</td>
<td>relocate, GENI</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Lau et al.</td>
<td>2002</td>
<td>Relocation from a holding list</td>
<td>exchange, relocate</td>
<td>Yes</td>
<td>Holding list for unrouted nodes, limit for number of routes</td>
</tr>
</tbody>
</table>
PERFORMANCE OF TABU SEARCH HEURISTICS FOR THE VRPTW

Average results with respect to Solomon’s benchmarks. The notations CNV and CTD in the rightmost column indicate the cumulative number of vehicles and cumulative total distance over all 56 test problems.

<table>
<thead>
<tr>
<th>Authors</th>
<th>R1</th>
<th>R2</th>
<th>C1</th>
<th>C2</th>
<th>RC1</th>
<th>RC2</th>
<th>CNV/CTD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Garcia et al.</td>
<td>12.92</td>
<td>3.09</td>
<td>10.00</td>
<td>3.00</td>
<td>12.88</td>
<td>3.75</td>
<td>436</td>
</tr>
<tr>
<td>(1994)</td>
<td>1317.7</td>
<td>1222.6</td>
<td>877.1</td>
<td>602.3</td>
<td>1473.5</td>
<td>1527.0</td>
<td>65977</td>
</tr>
<tr>
<td>Rochat et al.</td>
<td>12.25</td>
<td>2.91</td>
<td>10.00</td>
<td>3.00</td>
<td>11.88</td>
<td>3.38</td>
<td>415</td>
</tr>
<tr>
<td>(1995)</td>
<td>12085</td>
<td>961.72</td>
<td>828.4</td>
<td>589.9</td>
<td>1377.4</td>
<td>1119.6</td>
<td>57231</td>
</tr>
<tr>
<td>Potvin et al.</td>
<td>12.50</td>
<td>3.09</td>
<td>10.00</td>
<td>3.00</td>
<td>12.63</td>
<td>3.38</td>
<td>426</td>
</tr>
<tr>
<td>(1996)</td>
<td>1294.5</td>
<td>1154.4</td>
<td>850.2</td>
<td>594.6</td>
<td>1456.3</td>
<td>1404.8</td>
<td>63530</td>
</tr>
<tr>
<td>Taillard et al.</td>
<td>12.17</td>
<td>2.82</td>
<td>10.00</td>
<td>3.00</td>
<td>11.50</td>
<td>3.38</td>
<td>410</td>
</tr>
<tr>
<td>(1997)</td>
<td>1209.3</td>
<td>980.27</td>
<td>828.4</td>
<td>589.9</td>
<td>1389.2</td>
<td>1117.4</td>
<td>57523</td>
</tr>
<tr>
<td>Chiang et al.</td>
<td>12.17</td>
<td>2.73</td>
<td>10.00</td>
<td>3.00</td>
<td>11.88</td>
<td>3.25</td>
<td>411</td>
</tr>
<tr>
<td>(1997)</td>
<td>1204.2</td>
<td>986.32</td>
<td>828.4</td>
<td>591.4</td>
<td>1397.4</td>
<td>1229.5</td>
<td>58502</td>
</tr>
<tr>
<td>De Backer et al.</td>
<td>14.17</td>
<td>5.27</td>
<td>10.00</td>
<td>3.25</td>
<td>14.25</td>
<td>6.25</td>
<td>508</td>
</tr>
<tr>
<td>(1997)</td>
<td>1214.9</td>
<td>930.18</td>
<td>829.8</td>
<td>604.8</td>
<td>1385.1</td>
<td>1100.0</td>
<td>56998</td>
</tr>
<tr>
<td>Brandão (1999)</td>
<td>12.58</td>
<td>3.18</td>
<td>10.00</td>
<td>3.00</td>
<td>12.13</td>
<td>3.50</td>
<td>425</td>
</tr>
<tr>
<td></td>
<td>1205</td>
<td>995</td>
<td>829</td>
<td>591</td>
<td>1371</td>
<td>1250</td>
<td>58562</td>
</tr>
<tr>
<td>Schulze et al.</td>
<td>12.25</td>
<td>2.82</td>
<td>10.00</td>
<td>3.00</td>
<td>11.75</td>
<td>3.38</td>
<td>414</td>
</tr>
<tr>
<td>(1999)</td>
<td>1239.1</td>
<td>1066.7</td>
<td>828.9</td>
<td>589.9</td>
<td>1409.3</td>
<td>1286.0</td>
<td>60346</td>
</tr>
<tr>
<td>Tan et al. (2000)</td>
<td>13.83</td>
<td>3.82</td>
<td>10.00</td>
<td>3.25</td>
<td>13.63</td>
<td>4.25</td>
<td>467</td>
</tr>
<tr>
<td></td>
<td>1266.4</td>
<td>1080.2</td>
<td>870.9</td>
<td>634.8</td>
<td>1458.2</td>
<td>1293.4</td>
<td>62008</td>
</tr>
<tr>
<td>Lau et al. (2000)</td>
<td>14.00</td>
<td>3.55</td>
<td>10.00</td>
<td>3.00</td>
<td>13.63</td>
<td>4.25</td>
<td>464</td>
</tr>
<tr>
<td></td>
<td>1211.5</td>
<td>960.43</td>
<td>832.1</td>
<td>612.2</td>
<td>1385.0</td>
<td>1232.6</td>
<td>58432</td>
</tr>
<tr>
<td>Cordeau et al.</td>
<td>12.08</td>
<td>2.73</td>
<td>10.00</td>
<td>3.00</td>
<td>11.50</td>
<td>3.25</td>
<td>407</td>
</tr>
<tr>
<td>(2001)</td>
<td>1210.1</td>
<td>969.57</td>
<td>828.4</td>
<td>589.9</td>
<td>1389.8</td>
<td>1134.5</td>
<td>57556</td>
</tr>
<tr>
<td>Lau et al. (2002)</td>
<td>12.17</td>
<td>3.00</td>
<td>10.00</td>
<td>3.00</td>
<td>12.25</td>
<td>3.38</td>
<td>418</td>
</tr>
<tr>
<td></td>
<td>1211.5</td>
<td>1001.1</td>
<td>832.1</td>
<td>589.9</td>
<td>1418.8</td>
<td>1170.9</td>
<td>58477</td>
</tr>
</tbody>
</table>
RECENT WORK ON THE VRPTW

• Gehring and Homberger have proposed larger benchmark instances for the VRPTW (200-1000 customers)
• Several authors have presented methods for tackling these.
• Survey by Gendreau and Tarantilis almost completed.
REFERENCES

Introductory references on Local Search


Applications


Other references


1 Introduction

This chapter is a categorized bibliography of applications of metaheuristics for the Vehicle Routing Problem (VRP) and its extensions. It is basically a structured list of references based on the various metaheuristics and problem types.

2 Vehicle Routing Problem

The VRP [19] can be formally defined as follows. Let $G = (V,A)$ be a graph with $A$ the arc set and $V = \{1, \ldots, n\}$ the vertex set, where vertex 1 is the depot and the other vertices are cities or customers to be served. With every arc $(i,j)$, $i \neq j$, is associated a non-negative distance matrix $D = (d_{ij})$, where $d_{ij}$ can be interpreted either as a true distance, a travel time or a travel cost. Note that the undirected version of the VRP is obtained when $D$ is symmetric. A fleet of vehicles, based at the depot, is available for serving the vertices. Usually, the number of vehicles is variable, and a fixed cost $f$ is incurred each time a new vehicle is used. It can also happen that the number of vehicles is fixed or upper bounded. A non-negative weight or demand $q_i$ is associated with each vertex $i > 1$ and the sum of demands on any vehicle route should not exceed the vehicle capacity. The capacity and fixed cost can be the same for all vehicles (homogeneous fleet) or not (heterogeneous fleet). In some variants, the total travel distance or total travel time of each vehicle is also constrained. The problem is to find a set of least-cost vehicle routes such that:

- each vertex in $V - \{1\}$ is served exactly once by exactly one vehicle;
- each vehicle route starts and ends at the depot;
- all side constraints are satisfied (capacity, maximum travel distance or maximum travel time).

Note that this section also covers methods developed to solve Open VRP (OVRP), in which each route is a Hamiltonian path instead of Hamiltonian cycle; this difference comes from the fact that vehicles do not return to the starting depot or, if they do so, they must follow the same path backwards. Problems with multiple objectives are also considered.

The reader is referred to [9] for a general survey about metaheuristics for the classical VRP with capacity constraints. References on specific metaheuristics are found in the following subsections.

2.1 Ant colony optimization


### 2.2 Genetic algorithms


2.3 Greedy randomized adaptive search procedure


2.4 Simulated annealing


2.5 Tabu search


### 2.6 Variable neighborhood search


### 2.7 Others


3 VRP with Time Windows

In the VRP with Time Windows (VRPTW) [2], a time interval \([a_i, b_i]\) is associated with vertex \(i \in V\). In the hard time window variant, the vertex must be served within that interval (although the vehicle can wait, if it arrives before the lower bound \(a_i\)). In the soft time window variant, the vertex can be served outside of its time interval, but a penalty is incurred in the objective. A general survey about metaheuristics for the VRPTW is found in [1].

3.1 Ant colony optimization


3.2 Genetic algorithms


### 3.3 Greedy randomized adaptive search procedure

3.4 Simulated annealing


3.5 Tabu search


### 3.6 Variable neighborhood search


3.7 Others


## 4 VRP with Backhauls

In the VRP with Backhauls (VRPB) [20], the demand at each vertex $i$ corresponds either to a delivery or a pick-up (backhaul) which is then brought back to the depot. While goods are picked up or delivered, the quantity on board
should never exceed the capacity of the vehicle. This problem is a special case of the VRPPD (see Section 5).

### 4.2 Genetic algorithms


### 4.5 Tabu search


### 4.7 Others


### 5 VRP with Pick-ups and Deliveries

In the VRP with Pick-ups and Deliveries (VRPPD) [3], a transportation request $i$ is associated with two vertices $o_i$ and $d_i$, and the demand $q_i$ should be picked up at $o_i$ and delivered at $d_i$. For a solution to be feasible, both $o_i$ and $d_i$ should be in the same route. Furthermore, $o_i$ should appear before $d_i$ in the route. In this problem, capacity constraints can be present or not, depending on the application, and a time window is typically associated with each vertex.
For example, in transportation-on-demand applications where people with special needs are transported (a problem referred to as the Dial-A-Ride Problem), there are both capacity and time window constraints. Furthermore, there is a constraint on the maximum ride time of each passenger.

5.1 Ant colony optimization


5.2 Genetic algorithms


5.4 Simulated annealing


5.5 Tabu search


5.7 Others


### 6 VRP with Multiple Use of Vehicles

In standard vehicle routing problems, it is implicitly assumed that each vehicle serves a single route. In some cases, however, it might be possible or even necessary to assign the vehicle to several routes. This situation happens, for example, when the capacity of the vehicle is relatively small. In this case, frequent returns to the depot are required to load or unload the vehicle.

#### 6.2 Genetic algorithms


#### 6.5 Tabu search


### 7 Fleet Size and Mix VRP

When the number of vehicles is free and the fleet is heterogeneous, one is faced with the Fleet Size and Mix VRP (FSMVRP) [8], which exhibits special features that need to be addressed through specific algorithmic procedures. In particular, the benefits of replacing one type of vehicle by another for serving a particular route must be taken into account. We also include in this section methods
devised for solving the VRP with trailers (VRPT), where one has to determine the optimal deployment of a vehicle fleet of truck-trailer combinations.

7.2 Genetic algorithms

7.4 Simulated annealing


7.5 Tabu search


7.7 Others


8 VRP with Multiple Depots and Periodic VRP

In the VRP with Multiple Depots (MDVRP), there is not a single depot, but rather a number of depots with different locations and an associated fleet of vehicles. Depending on the variant considered, each vehicle may be required to terminate its route at its starting depot.

The Periodic VRP (PVRP) is an extension of the VRP in which customers must be visited one or more times during a planning horizon of several periods with routes performed by vehicles in each period. By substituting days for depots, one can show the equivalence of some variants of the MDVRP and the PVRP.

8.1 Ant colony optimization

8.2 Genetic algorithms


8.4 Simulated annealing


8.5 Tabu search


8.7 Others


9 **Dynamic VRP**

In dynamic vehicle routing problems [10, 17], some data about the problem are not known beforehand. That is, new information are revealed on-line, as the routes are executed by the vehicles. In most cases, a quick or real-time response time is also required. The new information often correspond to the occurrence of a new vertex (customer) that must be included into the current routes. It can also be some new information about the travel time of a vehicle, the current customer status (e.g., cancellation of a transportation request), etc. This section includes (repeats) papers on the dynamic variant of the VRPPD.

9.1 **Ant colony optimization**


9.2 **Genetic algorithms**


E. Taniguchi and H. Shimamoto. Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times. *Transportation*


### 9.5 Tabu search


### 9.7 Others

References


