TRAJECTORY-BASED SEARCH METHODS

Michel Gendreau

Département d'informatique et de recherche opérationnelle

and

Centre de recherche sur les transports

Université de Montréal

CP-AI-OR'05 Master Class
Prague, May 29, 2005

ORGANIZATION OF THE TUTORIAL

1. Introduction
2. Neighbourhoods and search spaces
3. Main classes of trajectory based search methods
4. Tabu Search
5. Recent trends in Tabu and Local Search
6. Tricks of the trade
7. References

There will be a short question period after each part of the tutorial.

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)* = {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
- 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 (λ-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

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

$$\text{subject to } \sum_{j \in J} x_{ij} = d_i, \; i \in I$$

$$\sum_{i \in J} x_{ij} \leq K_j y_j, \; j \in J$$

$$x_{ij} \geq 0, \; i \in I, \; j \in J$$

$$y_j \in \{0,1\}, \; j \in J$$

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 $\tilde{\boldsymbol{y}}$ of location variables, optimal (w.r.t. to this plant configuration) values for the flow variables $\boldsymbol{x}(\tilde{\boldsymbol{y}})$ can be retrieved by solving the associated transportation problem:

$$\text{(TP)} \quad \text{Minimize } z(\tilde{\boldsymbol{y}}) = \sum_{i \in I} \sum_{j \in J} c_{ij}\, x_{ij}$$

$$\text{subject to} \sum_{j \in J} x_{ij} = d_i,\ i \in I$$

$$\sum_{i \in J} x_{ij} \le K_j \tilde{y}_j,\ j \in J$$

$$x_{ij} \ge 0,\ \ i \in I\ ,\ j \in J$$

If $\tilde{\boldsymbol{y}} = \boldsymbol{y}^*$, the optimal location vector, the optimal solution to the original CPLP problem is simply given by $(\boldsymbol{y}^*, \boldsymbol{x}(\boldsymbol{y}^*))$.

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:

$$\sum_{j \in J} x_{ij} = d_i, \; i \in I$$

$$\sum_{i \in J} x_{ij} \leq K_j, \; j \in J$$

$$x_{ij} \geq 0, \; i \in I \; , \; j \in J$$

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 "neigbourhood" 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*

- $S :\in \underset{S' \in N(S)}{\arg\max} \; [f(S')]$;
- *if* $f(S) \rangle f^*$, *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 <—> Configurations of particles
- Objective function <—> 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

- Introduced, by Hansen and Mladenović in 1997.
- 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,ℓ) 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 "neigbourhood" 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 $S := S_0$, $f^* := f(S_0)$, $S^* := S_0$, $T := \varnothing$

Search

While termination criterion not satisfied *do*

- $S :\in \underset{S' \in \overline{N}(S)}{\arg\max}\ [f(S')]$;
- *if* $f(S) \rangle f^*$, *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;...).

  $\rightarrow$ 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.

Taxonomy paper by Crainic, Toulouse and Gendreau (1997).

Book edited by E. Alba (2005).

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)
  - Lopez, Carter and Gendreau (1998)

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.

TRICKS OF THE TRADE

GETTING STARTED

- **Read** one or two good introductory papers (to gain some knowledge of the concepts, of the vocabulary,...).

- **Read** several papers describing in detail applications in various areas (to see how concepts are implemented).

- **Think** a lot about your problem

  $\rightarrow$ on **search space**
  and **neighbourhood structure** .

- **Implement a simple** version of LS or TS based on that search space and this neighbourhood.

- **Collect statistics** on the performance of your simple heuristic.

  $\rightarrow$ memories (recency, frequency,...)

- **Analyze results** and **adjust** the algorithm

  $\rightarrow$ add diversification, intensification, ...

I have implemented a tabu search heuristic, but I keep on getting mediocre results. What should I do now? (HELP!)

- If there are constraints, consider **penalization** to «open up» the search.

- Change the **neighbourhood structure** to allow for a more purposeful evaluation of moves.

- **Collect statistics**

- **Follow the algorithm step by step**

- Consider **diversification**

- Change parameter values

How do I calibrate all those parameters?
How do I go about computational testing?

- Get a good set of test problems preferably with some measures of problem difficulty (estimated beforehand).

- Split your problem set into two subsets:

  - one for algorithmic design and parameter calibration,

  - the other to perform your final computational testing (to be published).

- Perform exploratory testing to find good ranges of parameters.

- Fix the values of «robust» parameters.

- Perform systematic testing for other parameters.

- You may read the Crainic, Gendreau, Soriano and Toulouse paper in *Annals of O.R*. **41**.

I use probabilistic tabu search. My results are fairly good, but when I look at my solutions, they look somewhat strange!?!

Are you sure that your solutions are local optima w.r.t your inner heuristic?

They may very well not be, unless you do something about it!

- Perform a “straight” local improvement phase, starting from the best found solution, at the end of the TS.

- Switch to TS without sampling (again from the best found solution) for a short duration before completing the algorithm.

REFERENCES

Introductory

Aarts, E. and J.K. Lenstra (eds.) (2003), *Local Search in Combinatorial Optimization*, Wiley, Chichester.

Gendreau, M. (2003), "An Introduction to Tabu Search", in *Handbook of Metaheuristics*, F.W. Glover and G.A. Kochenberger (eds.), Kluwer, Boston, MA, 37-54.

Glover, F., É. Taillard and D. de Werra (1993), "A User's Guide to Tabu Search", *Annals of O.R.* **41**, 3-28.

Glover, F. and M. Laguna (1993), "Tabu Search", in *Modern Heuristic Techniques for Combinatorial Problems*, C.R. Reeves (ed.), Blackwell, 70-150.

Glover, F. and M. Laguna (1997), *Tabu Search*, Kluwer.

Hertz, A. and D. de Werra (1991), "The Tabu Search Metaheuristic: How We Used It", *Annals of Mathematics and Artificial Intelligence* **1**, 111-121.

Soriano, P. and M. Gendreau (1997), « Fondements et applications des méthodes de recherche avec tabous», *RAIRO* (*Recherche opérationnelle*) **31**, 133-159.

Applications

de Werra, D., F. Glover, M. Laguna, É. Taillard (eds.) (1993), *Annals of Operations Research* **41**, "Tabu Search".

Laporte, G. and I. Osman (eds.) (1996), *Annals of Operations Research* **63**, "Metaheuristics in Combinatorial Optimization".

Osman, I.H. and J.P. Kelly (eds.) (1996), *Meta-Heuristics: Theory and Applications*, Kluwer Academic Publishers, Norwell, MA.

Ribeiro, C.C. and P. Hansen (eds.) (2002), *Essays and Surveys in Metaheuristics*, Kluwer Academic Publishers, Norwell, MA.

Voss, S., S. Martello, I.H. Osman and C. Roucairol (eds.) (1999), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publishers, Norwell, MA.

Other references

Battiti, R. and G. Tecchiolli (1994), "The Reactive Tabu Search", *ORSA Journal on Computing* **6**, 126-140.

Bräysy, O. and M. Gendreau (2002), "Tabu Search Heuristics for the Vehicle Routing Problem with Time Windows", *TOP* **10**, 211-237.

Crainic, T.G. and M. Gendreau (1999), "Towards an Evolutionary Method – Cooperative Multi-Thread Parallel Tabu Search Heuristic Hybrid", in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, S. Voss, S. Martello, I.H. Osman and C. Roucairol (eds.), Kluwer Academic Publishers, pp. 331-344.

Crainic, T.G. and M. Gendreau (2002), "Cooperative Parallel Tabu Search for Capacitated Network Design", *Journal of Heuristics* **8**, 601-627.

Crainic, T.G., M. Gendreau and J.M. Farvolden (2000) "Simplex-based Tabu Search for the Multicommodity Capacitated Fixed Charge Network Design Problem", *INFORMS Journal on Computing* **12**, 223-236.

Crainic, T.G., M. Gendreau, P. Soriano and M. Toulouse (1993), "A Tabu Search Procedure for Multicommodity Location/Allocation with Balancing Requirements", *Annals of Operations Research* **41**, 359-383.

Crainic, T.G., M. Toulouse and M. Gendreau (1997), "Toward a Taxonomy of Parallel Tabu Search Heuristics", *INFORMS Journal on Computing* **9**, 61-72.

Cung, V.-D., S.L. Martins, C.C. Ribeiro and C. Roucairol (2002), "Strategies for the Parallel Implementation of Metaheuristics", in *Essays and Surveys in Metaheuristics*, C.C. Ribeiro and P. Hansen (eds.), Kluwer Academic Publishers, pp. 263-308.

Dueck, G. (1993), "New optimization heuristics: The great deluge algorithm and record-to-record travel", *Journal of Computational Physics* **90**, 161–175.

Dueck, G. and T. Scheurer (1990), "Threshold accepting: A general purpose optimization algorithm", *Journal of Computational Physics* **104**, 86–92.

Fleurent, C. and J.A. Ferland (1996), "Genetic and Hybrid Algorithms for Graph Colouring", *Annals of Operations Research* **63**, 437-461.

Gendreau, M. (2002), "Recent Advances in Tabu Search", in *Essays and Surveys in Metaheuristics*, C.C. Ribeiro and P. Hansen (eds.), Kluwer Academic Publishers, pp. 369-377.

Gendreau, M., F. Guertin, J.-Y. Potvin and É.D. Taillard (1999), "Parallel Tabu Search for Real-Time Vehicle Routing and Dispatching", *Transportation Science* **33**, 381-390.

Gendreau, M., A. Hertz and G. Laporte (1994), "A Tabu Search Heuristic for the Vehicle Routing Problem", *Management Science* **40**, 1276-1290.

Gendreau, M., G. Laporte and J.-Y. Potvin (2002), "Metaheuristics for the Capacitated VRP", in *The Vehicle Routing* Problem, P. Toth and D. Vigo (eds.), SIAM Monographs on Discrete Mathematics and Applications, pp. 129-154.

Gendreau, M., P. Soriano and L. Salvail (1993), "Solving the Maximum Clique Problem Using a Tabu Search Approach", *Annals of Operations Research* **41**, 385-403.

Glover, F. (1977), "Heuristics for Integer Programming Using Surrogate Constraints", *Decision Sciences* **8**, 156-166.

Glover, F. (1986), "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers and Operations Research* **13**, 533-549.

Glover, F. (1989), "Tabu Search – Part I", *ORSA Journal on Computing* **1**, 190-206.

Glover, F. (1990), "Tabu Search – Part II", *ORSA Journal on Computing* **2**, 4-32.

Glover, F. (1992), "Ejection chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems", University of Colorado. Shortened version published in *Discrete Applied Mathematics* **65**, 223-253, 1996.

Grünert, T. (2002), "Lagrangean Tabu Search", in *Essays and Surveys in Metaheuristics*, C.C. Ribeiro and P. Hansen (eds.), Kluwer Academic Publishers, pp. 379-397.

Hansen, P. and N. Mladenović (1997), "Variable Neighbourhood Search for the *p*-Median", *Location Science* **5**, 207–226.

Kirkpatrick, S., C.D. Gelatt Jr. and M.P. Vecchi (1983), "Optimization by Simulated Annealing", *Science* **220**, 671-680.

Lokketangen, A. and F. Glover (1996), "Probabilistic Move Selection in Tabu Search for 0/1 Mixed Integer Programming Problems", in *Meta-Heuristics: Theory and Applications*, I.H. Osman and J.P. Kelly (eds.), Kluwer Academic Publishers, pp. 467-488.

Osman, I.H. (1993), "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem", *Annals of Operations Research* **41**, 421-451.

Pesant, G. and M. Gendreau (1999), "A Constraint Programming Framework for Local Search Methods", *Journal of Heuristics* **5**, 255-280.

Rego, C. and C. Roucairol (1996), "A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem", in *Meta-Heuristics: Theory and Applications*, I.H. Osman and J.P. Kelly (eds.), Kluwer Academic Publishers, pp. 661-675.

Rochat, Y. and É.D. Taillard (1995), "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing", *Journal of Heuristics* **1**, 147-167.

Rolland, E. (1996), "A Tabu Search Method for Constrained Real-Number Search: Applications to Portfolio Selection", Working Paper, The Gary Anderson Graduate School of Management, University of California, Riverside.

Skorin-Kapov, J. (1990), "Tabu Search Applied to the Quadratic Assignment Problem", *ORSA Journal on Computing* **2**, 33-45.

Soriano, P. and M. Gendreau (1996), "Diversification Strategies in Tabu Search Algorithms for the Maximum Clique Problems", *Annals of Operations Research* **63**, 189-207.

Taillard, É. (1990), "Some efficient heuristic methods for the flow shop sequencing problem", *European Journal of Operational Research* **47**, 65-74.

Taillard, É. (1991), "Robust taboo search for the quadratic assignment problem", *Parallel Computing* **17**, 443-455.

Taillard, É.D., P. Badeau, M. Gendreau, F. Guertin and J.-Y. Potvin (1997), "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows", *Transportation Science* **31**, 170−186.