World Academy of Science, Engineering and Technology
International Journal of Transport and Vehicle Engineering
Vol:7, No:1, 2013

How to Build and Evaluate a Solution Method: an Illustration for the Vehicle Routing Problem

Nicolas Zufferey

***Abstract*—**The vehicle routing problem (VRP) is a famous combinatorial optimization problem. Because of its well-known difficulty, metaheuristics are the most appropriate methods to tackle large and realistic instances. The goal of this paper is to highlight the key ideas for designing VRP metaheuristics according to the following criteria: efficiency, speed, robustness, and ability to take advantage of the problem structure. Such elements can obviously be used to build solution methods for other combinatorial optimization problems, at least in the deterministic field.

***Keywords*—**Vehicle routing problem, Metaheuristics, Combinatorial optimization.

I. Introduction

THE *vehicle routing problem* (VRP) was introduced in [1] and can be described as follows. A fleet of m identical vehicles of capacity Q is based at the depot v_0, where m is known or could also be a decision variable. Let G be a graph with vertex set $\{v_0, v_1, \ldots, v_n\}$ where the v_i's (for $i > 0$) are the customers. For each i, j in $\{0, 1, \ldots, n\}$, the cost c_{ij} and travel time t_{ij} are known. Usually, the cost and travel time matrices are symmetrical, thus edges instead of arcs can be considered between the vertices. With each customer i is associated a demand q_i and a service time t_i. The VRP consists in building at most m routes such that:

- Each route starts and ends at the depot,
- Each customer is visited exactly once by exactly one vehicle,
- The total demand of each route does not exceed Q,
- The total duration of each route (including travel and service times) does not exceed D,
- The total routing cost is minimized.

Several extensions have been studied. Currently, no known exact method is able to solve all instances with up to 50 vertices [2]. For survey papers on the VRP, the reader is referred to [3] – [8]. If $m = 1$ and $Q = D = +\infty$, then the VRP reduces to the famous *traveling salesman problem*. The VRP is NP-hard and among the most studied in the combinatorial optimization community. Because of its well-known difficulty, metaheuristics are the most appropriate methods to tackle medium and large instances.

Modern methods for solving complex optimization problems are often divided into *exact* methods and *metaheuristic* methods. An exact method guarantees that an optimal solution will be obtained in a finite amount of time. Among the exact methods are branch-and-bound, dynamic programming, Lagrangian relaxation based methods, and linear and integer programming based methods (e.g., branch-and-cut, branch-and-price, branch-and-cut-and-price) [9]. However, for a large number of applications and most real-life optimization problems, which are typically NP-hard [10], such methods need a prohibitive amount of time to find an optimal solution. For these difficult problems, it is preferable to find a satisfying solution in a reasonable amount of time. This is exactly the goal of *heuristics* (simple solution methods) and *metaheuristics* (more advanced solution methods).

For a survey on metaheuristics, the reader is referred to [11]. There mainly exist two classes of metaheuristics: *local search methods* (e.g., simulated annealing, tabu search, variable neighborhood search, guided local search, threshold algorithms, GRASP), and *population based methods* (e.g., genetic algorithms, ant colonies, scatter search, adaptive memory algorithms and memetic search which can be seen as a generalization of genetic algorithms).

A local search method starts with an initial solution and tries to improve it iteratively. At each iteration, a modification of the current solution, called a *move*, is performed in order to generate a *neighbor* solution. The definition of a move, i.e. the definition of the neighborhood structure, depends on the problem considered. In order to prevent cycling, in a tabu search [12], the reverse of the recently performed moves are forbidden for a few iterations.

In contrast, *population based methods*, also called *evolutionary algorithms*, can be defined as iterative procedures that use a central memory to store and operate on certain solutions collected during the search process. Each iteration, called a *generation*, involves two complementary ingredients: cooperation and self-adaptation. In the *cooperation* effort, the central memory is used to build new offspring solutions, while *self-adaptation* consists of individually modifying the offspring solutions. The output solutions of the self-adaptation phase are used for updating the content of the central memory. In some population based algorithms, the self-adaptation phase can be performed with the use of a local search which is applied individually to some of the offspring solutions. For simplicity, such algorithms are also classified in population based methods in this paper.

Theoretically, there exist some convergence theorems associated with the use of metaheuristics (e.g., [13] – [16]). Basically, the theorems state that the search has a high probability to find an optimal solution, but in a huge amount of time, which is likely to be larger than the time needed for a complete enumeration. Therefore, such theoretical results do not have any impact in practice, and, moreover, do not help to efficiently design a metaheuristic.

The performance of a metaheuristic can be evaluated according to several criteria, and not only its *efficiency* (i.e. the quality of the obtained results). The three criteria studied in

N. Zufferey is with HEC – University of Geneva, Switzerland (phone: +41-22-379-8124; e-mail: nicolas.zufferey-hec@unige.ch).

this paper are: *speed* (i.e. time needed to get good results), *robustness* (i.e. sensitivity to variations in problem characteristics and data quality), and *ability to take advantage of problem structure*. It is usually difficult to design a metaheuristic having a good behavior according to all these criteria.

The goal of this paper, which is based on [17] and specifically focuses on the VRP, is to highlight the key ideas when designing metaheuristics for the VRP, according to speed (Section II), robustness (Section III), and the ability to take advantage of the problem structure (Section IV). It is assumed that all the presented ideas contribute to efficiency.

II. Ideas for the Design of Quick Solution Methods

In this section are presented key ideas which are useful when designing quick metaheuristics for the VRP.

A. Generation of a Single Solution

The generation of a single solution in the selected solution space should not be time consuming; if necessary, relax some constraints in order to deal with non feasible solutions, allowing the objective function to be adjusted to achieve this goal.

Sometimes, even the goal of finding a feasible solution for the problem is NP-hard. In such a case, one can relax some constraints and add a weighted penalty term $\alpha \bullet P(s)$ to the objective function, which penalizes any constraint violation occurring in solution s. The idea is to choose an arbitrary initial value for α, and then: to augment it if the search does not often encounter feasible solutions, and to reduce it if the search seems to be trapped in a region of the solution space only containing feasible solutions [18].

For the VRP, if the number m of vehicles is small when compared to the number n of customers, it can be difficult to generate a feasible solution with respect to the given values of Q (maximum capacity of a vehicle) and D (maximum route duration associated with a vehicle). As proposed in the efficient tabu search proposed in [18] and called TABUROUTE, two penalty terms can be added to the objective function, which are the capacity and the route duration violations.

B. Reduction of the Solution Space

A solution method is obviously quicker if the solution space is reduced, that is if the set of values that each variable can have is reduced. When no loss of generality is entailed, the reduction of the search space can be done by the use of *arc-consistency* techniques (e.g., [19] and [20]), where some inconsistent values are removed from the set of possible values of some variables. Another approach consists in solving a subproblem of the general problem (with an exact method or a heuristic), in order to reduce the problem size.

For the VRP, in [21], the author proposed a decomposition of the problem in smaller subproblems. This allows the use of parallel processors and strongly reduces the computing time, because each subproblem can be solved independently (even if periodic moves of vertices to adjacent sectors are necessary).

In the granular tabu search proposed in [22], unpromising edges are removed from the graph, i.e. edges which have a small chance of being contained in an optimal solution. Thus, the remaining graph becomes sparse and the search is significantly accelerated.

C. Simplicity

If two methods obtain the same kind of results within the same amount of time, the simpler one is obviously better. As mentioned in [4]: "simple codes, preferably short and self-contained, stand a better chance of being adopted, although a minimum of complexity is to be expected for good results".

For the VRP, simple and straightforward solution methods are already discussed in this paper (such as λ-exchanges moves with small values of λ, as presented in subsection II.D). In contrast, a rather complex tabu search method, with numerous parameters, was proposed in [23], which is not competitive with the best algorithms. It uses sophisticated moves: the authors consider *swaps* of vertices between routes, *repositioning* of vertices into other routes (based on the search of an optimal flow in a network), and improvements on local routes (based on *3-opt* exchanges, as developed in [24]).

D. Conservation of the Solution Structure

When designing local search methods, slight moves should be used. Thus, any neighbor solution should keep the major part of the current solution. As a consequence, a guided search in the solution space would be possible: we know from where we come and we know where we go, and no big jump is performed over the solution space.

For the VRP, in *λ-exchanges* proposed in [25], a move consists in exchanging at most λ customers between two routes (where typically λ is in {1, 2}). This includes simple *swap* moves (i.e. two vehicles exchange a customer) and simple *insertion* moves (i.e. a customer is moved from a route to another one). λ-exchanges moves (with small values of λ) preserve the solution structure, as it is for example the case in the tabu search proposed in [21], where standard vertex insertions and exchanges are performed.

In contrast, the first implementation of simulated annealing [26] did not provide competitive results, probably because of its neighborhood structure: a move involves several ingredients, such as reversing part of a route, moving part of a route within the same route, and trading vertices between two routes.

In addition, *ejection chain* moves [27] do also not preserve the major part of the current solution structure, as a move consists in moving vertices in a cyclic manner from route 1 to route 2, from route 2 to route 3, and so on.

E. Incremental Computation

A neighbor solution should be evaluated from the current solution, its value and the considered move. Formally, let s be the current solution and s' a neighbor solution obtained by performing move m from s. It should be possible to compute $f(s')$ from s, m and $f(s)$. More precisely, it should be possible to evaluate how much m deteriorates $f(s)$ and how much m improves $f(s)$, which is respectively denoted $lost(m)$ and

gain(m). Thus, assuming *f* has to be minimized, we have *f(s') = f(s) + lost(m) − gain(m)*, which is usually much quicker to compute than *f(s')* from scratch.

For the VRP, incremental computation is easy to develop for *insertion* moves or *swap* moves, because the evaluation only focuses on the two involved routes. In contrast, an efficient incremental computation is difficult to design for *ejection chains*.

III. Ideas for the Design of Robust Solution Methods

In this section are presented key ideas which are useful when designing robust metaheuristics for the VRP.

A. Connectivity of the Solution Space

A local search should use *flexible* moves so that from any solution *s*, it should be possible to reach any other solution *s'* of the solution space by performing a sequence of moves. Algorithms in which an optimal solution can be reached from any starting solution, and in which the probability of selecting a given move from a current solution is bounded below by a positive constant, are referred to in [28] as probabilistically approximately complete. Such algorithms have the characteristic that the probability to find an optimal solution approaches 1 if the computation time approaches ∞.

For the VRP, two main ways are possible to define neighbor solutions. Moves based on *λ-exchanges* (with small values of *λ*) satisfy the *connectivity* principle. In contrast, moves based on *ejection chains* do not seem to satisfy it. Note that local search methods based on ejection chains do not outperform algorithms based on *λ*-exchanges [6].

B. Escape from Local Optimum

A local search should be able to escape from any local optimum. A local optimum is defined according to a specific neighborhood structure *N*. More precisely, solution *s* is a *local optimum* for *N* if there is no better solution in *N(s)*, i.e. in the set of its neighbor solutions. Thus, a solution *s* could be a local optimum for one neighborhood structure but not for another. To escape from a local optimum, simulated annealing has an acceptation probability [29], tabu search has a tabu list [12], and variable neighborhood search uses various neighborhood structures with different amplitudes [30].

For the VRP, for *insertion* moves, tabu status prevent cycling by forbidding the reinsertion of a vertex in its original route, or by forbidding to move again a vertex if it was just moved from one route to another.

C. Diversity of the Population

An evolutionary algorithm should preserve the diversity of the involved population. To reach this goal, a *diversity measure* can be used. The mechanism which should preserve the diversity of the population is usually the self-adaptation phase of every population based method. It is for example the greedy force in ant colonies and the mutation operator in genetic algorithms. It is commonly recognized that the variety of genes in the population is a crucial aspect of performance in genetic algorithms. As mentioned in [31], the process of convergence can be described by measuring the disorder of genes in terms of the entropy *ENT*, which is ideally 1 for the initial population with randomly generated solutions, and 0 at the end of the search when the population converged to a global optimum.

For the VRP, the memetic algorithm proposed in [32] is based on the *permutation* approach, where a solution is represented with a permutation of all the customers (without the depot), without any assignment to the vehicles. An exact method (based on the shortest path problem in an auxiliary graph) is then used to optimally assign the customers to the vehicles. A crossover is then defined as follows: two cutting places *i* and *j* are first chosen in the first parent, then the corresponding string is placed in positions *i, ... , j* in the offspring solution, then the second parent is circularly swept from position *j + 1* in order to complete the offspring solution. The way to represent a solution, and such a crossover, strongly help in preserving the diversity of the population.

D. Diversification of the Search

A solution method should be able to explore solutions far away from those already visited.

On the one hand, when considering local search techniques, the *distance* between two solutions can be measured by the minimum number of moves which are necessary to transform one solution to the other. The associated computation is however intractable in most cases. Therefore, distances which are independent of the moves are generally more relevant. Such distance functions can help in guiding the search towards new and promising areas of the solution space.

On the other hand, in most of the population based methods, the cooperation phase has a diversification role: the popular *one-point* or *two-point* or *uniform* crossovers used in genetic algorithms lead to the generation of an offspring solution which is usually very different from its parent solutions.

For the VRP, in [18], a diversification strategy consists of adding a term to the objective function, which penalizes the movement of specific vertices (from their position in their route) which were frequently moved during previous search. This encourages other vertices to be moved, and thus favors the exploration of new parts of the solution space.

IV. Taking Advantage of the Problem Structure

In this section are presented key ideas which are useful when designing VRP metaheuristics which are able to take advantage of the problem structure.

A. Choice of the Solution Space

The representation of a solution should incorporate the properties of the problem. For example, the most efficient genetic algorithms do not use binary vectors to represent a solution, but are based on an encoding that undertakes to capture essential features of the problem structure. Once a solution representation has been selected, the solution space of the problem can be defined, which is the set of all the solutions of the problem.

For the VRP, the most straightforward way to represent a solution is to indicate, for each vehicle, the sequence of the visited customers. In other words, for each vehicle, its *route* is known. In contrast, the memetic algorithm proposed in [32] is based on the permutation approach (as described in subsection III.D). Even if it is a non-natural representation, it is helpful for the design of a specialized crossover operator.

B. Use Repairing Moves

A local search should use repairing moves: at least a drawback of the current solution should be removed from it when generating the neighbor solution, even if several new drawbacks are created in the latter. It also helps to escape from local optima and to increase the robustness of the method. If repairing moves are difficult to design (because it is difficult to locate a drawback in the solution), an idea is to use moves which do not deteriorate too much the quality of the current solution, based on the properties of the considered problem.

For the VRP, a powerful neighborhood structure is proposed in TABUROUTE, where a move consists in putting a vertex x in a new route, which contains one of the closest vertex to x, by means of an efficient generalized insertion procedure called GENI [33].

C. Handle Relevant Information along the Search Process

The nature of the information transmitted during the cooperation phase of an evolutionary algorithm should be based on the properties of the considered problem. In general, as mentioned in [34], the crossover operator is regarded as a main genetic operator and the performance of the genetic algorithms depends on the performance of the used crossover operator. Specialized crossovers have been successfully proposed for genetic algorithms for various problems, which lead to much better results than the ones obtained with uniform, 1-point or 2-points crossovers.

For the VRP, in the adaptive memory algorithm proposed in [35] (an *adaptive memory algorithm* can be seen as a generalization of a genetic algorithm, where the recombination operator is not limited to two parents and a local search procedure is used for the self-adaptation phase), an offspring solution is built route by route by taking them from the central memory M, while giving more chance to good routes to be selected (the value of a route is defined according to the value of the solution it belongs). While taking routes, an effort is made to avoid selecting routes with customers belonging to the current offspring solution. At the end of the process, a tabu search procedure builds a solution with the selected routes from M and the remaining unassigned customers. In contrast, the adaptive memory proposed in [36] does not combine full vehicle routes, but route segments (called bones) taken from good quality routes.

On the contrary to the two recombination operators proposed in [35] and [36], the trail systems proposed in all ant algorithms is very limited and do not take advantage of the nature of the problem, because only pairs of vertices are considered (instead of longer segments or routes) as follows: if the previous « good » ants visited customers i and j successively, this information should be transmitted to the next generations of ants.

D. Guided Cooperation Phase

The cooperation phase of an evolutionary algorithm should not be equivalent to a restart procedure: an offspring solution should have relevant information from the past history.

For the VRP, this principle is satisfied in the recombination proposed in the adaptive memory algorithm developed in [35], where full routes are copied from the central memory to generate an offspring solution. In contrast, the existing ant algorithms do not seem to satisfy this principle, because the role of each ant is to build a solution from scratch, based on a trail system only involving pairs of vertices.

V. Conclusion

In this paper are presented various elements which are helpful for designing solution methods for combinatorial optimization problems. The discussion is illustrated for the vehicle routing problem for which numerous metaheuristics have been proposed. Some of these metaheuristics are discussed according to various criteria which can measure their quality: speed, robustness, and the ability to take advantage of the problem structure. Despite of their relevance, these three criteria are often underestimated when analyzing the obtained results of an algorithm. It is difficult to develop a metaheuristic which has a good performance according to all the performance criteria. However, for a given optimization problem, a given computation time limit, and a given deadline to develop a solution method, the elements proposed in this paper should help in the selection of the metaheuristic and the ingredients that will be incorporated in it.

References

[1] Dantzig, G. B. and Ramser, J. H. 1959. The truck dispatching problem. **Management Science** 6, 80–91.
[2] Toth, P. and Vigo, D. 1998a. Fleet Management and Logistics. Kluwer, Boston, Chapter Exact solution of the vehicle routing problem, 1–31.
[3] Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I.-M. 1998. Fleet Management and Logistics. Kluwer, Boston, Chapter Metaheuristics in vehicle routing, 33–56.
[4] Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., and Semet, F. 2002. A Guide to Vehicle Routing Heuristics. **Journal of the Operational Research Society** 53 (5), 512–522.
[5] Laporte, G. and Semet, F. 2002. The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, Chapter Classical heuristics for the capacitated VRP, 109–128.
[6] Gendreau, M., Laporte, G., and Potvin, J.-Y. 2002. The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, Chapter Metaheuristics for the VRP, 129–154.
[7] Cordeau, J.-F. and Laporte, G. 2004. Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search. Kluwer, Boston, Chapter Tabu search heuristics for the vehicle routing problem, 145–163.
[8] Cordeau, J.-F., Gendreau, M., Hertz, A., Laporte, G., and Sormany, J.-S. 2005. Logistics Systems: Design and Optimization. Springer, Chapter New Heuristics for the Vehicle Routing Problem, 270–297.
[9] Nemhauser, G. and Wolsey, L. 1988. Integer and Combinatorial Optimization. John Wiley & Sons.
[10] Garey, M. and Johnson, D. 1979. Computer and Intractability: a Guide to the Theory of NP-Completeness. Freeman, San Francisco.
[11] Gendreau, M., Potvin, J.-Y., Handbook of Metaheuristics, Springer, 2010.

[12] Glover, F. 1989. Tabu search - part I. **ORSA Journal on Computing** 1, 190–205.
[13] Aarts, E. H. I. and Laarhoven, P. J. M. 1985. Statistical cooling: A general approach to combinatorial optimization problems. **Philips Journal of Research** 40, 193–226.
[14] Hajek, B. 1988. Cooling schedules for optimal annealing. **Mathematics of Operations Research** 13, 311–329.
[15] Faigle, U. and Kern, W. 1992. Some convergence results for probabilistic tabu search. **ORSA Journal on Computing** 4, 32–37.
[16] Glover, F. and Hanafi, S. 2002. Tabu Search and Finite Convergence. **Discrete Applied Mathematics** 119 (1–2), 3–36.
[17] N. Zufferey, *Metaheuristics: some Principles for an Efficient Design*, **Computer Technology and Applications** 3 (6), 446–462, 2012
[18] Gendreau, M., Hertz, A., and Laporte, G. 1994. A tabu search heuristic for the vehicle routing problem. **Management Science** 40, 1276–1290.
[19] Mohr, R. and Henderson, T. C. 1977. Arc and Path Consistency Revisited. **Artificial Intelligence** 28, 225–233.
[20] Bessière, C. 1994. Arc-Consistency and Arc-Consistency Again. **Artificial Intelligence** 65, 179–190.
[21] Taillard, E. D. 1993. Parallel iterative search methods for vehicle routing problems. **Networks** 23, 661–673.
[22] Toth, P. and Vigo, D. 1998b. The granular tabu search (and its application to the vehicle routing problem). Tech. Rep. OR-98-9, DEIS - Univertity of Bologna - Italy.
[23] Xu, J. and Kelly, J. 1996. A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem. **Transportation Science** 30, 379–393.
[24] Lin, S. 1965. Computer solutions of the traveling salesman problem. **Bell System Technical Journal** 44, 2245–2269.
[25] Osman, I. H. 1993. Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem. **Annals of Operations Research** 41, 421–451.
[26] Robusté, F., Daganzo, C. F., and Souleyrette, R. 1990. Implementing vehicle routing models. **Transportation Research, Part B: methodological** 24 (4), 263–286.
[27] Rego, C. and Roucairol, C. 1996. Meta-Heuristics: Theory and Applications. Kluwer, Boston, Chapter A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem, 661–675.
[28] Hoos, H. H. 1998. Stochastic Local Search - Methods, Models, Applications. Ph.D. thesis, Darmstadt University of Technology.
[29] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. 1983. Optimization by simulated annealing. **Science** 220 (5498), 671–680.
[30] Mladenovic, N. and Hansen, P. 1997. Variable neighborhood search. **Computers & Operations Research** 24, 1097–1100.
[31] Mattfeld, D. C, Bierwirth, C., and Kopfer H. 1999. A search space analysis of the job shop scheduling problem. **Annals of Operations Research** 86, 441-453.
[32] Prins. 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. **Computers & Operations Research** 31, 1985–2002.
[33] Gendreau, M., Hertz, A., and Laporte, G. 1992. New insertion and postoptimization procedures for the traveling salesman problem. **Operations Research** 40, 1086–1094.
[34] Cheng, R., Gen, M., and Tsujimura, Y. 1999. A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies. **Computers & Industrial Engineering** 36, 343–364.
[35] Rochat, Y. and Taillard, E. 1995. Probabilistic diversification and intensification in local search for vehicle routing. **Journal of Heuristics** 1, 147–167.
[36] Tarantilis, C.-D. and Kiranoudis, C. T. 2002. Bone Route: An adaptive memory-based method for effective fleet management. **Annals of Operations Research** 115, 227–241.