European Society of Computational Methods in Sciences and Engineering (ESCMSE)

Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM)
vol. 9-10, no. 3-4, 2016, pp. 57-67
ISSN 1790–8140

Unconventional Heuristics for Vehicle Routing Problems

Eva Volna and Martin Kotyrba[1]

Faculty of Science
Department of Informatics and Computers,
University of Ostrava,
30 dubna 22, 70103 Ostrava, Czech Republic

Received; accepted in revised form

Abstract: The Vehicle Routing Problem (VRP) is one of the most challenging combinatorial optimization tasks. This problem consists in designing an optimal set of routes for a fleet of vehicles in order to serve a given set of customers. Vehicle routing problem forms an integral part of the supply chain management, which plays a significant role in productivity improvement in organizations through an efficient and effective delivery of goods/services to customers. This problem is known to be NP-hard; hence many heuristic procedures for its solution have been suggested. For such problems, it is often desirable to obtain approximate solutions, so they can be found fast enough and are sufficiently accurate for the purpose. In this paper, we have performed an experimental study that indicates a suitable use of genetic algorithms for the vehicle routing problem. We tested instances from Capacitated Vehicle Routing Problem Library (CVRPLIB) series A, B, E, M and X. The obtained experimental outputs were compared with the following heuristics: the Clarke and Wright heuristic, sweep algorithm, and Taillard's algorithm.

Keywords VRP - Vehicle Routing Problem, combinatorics, Clarke and Wright heuristic, Sweep algorithm, Taillard's algorithm, Genetic algorithm.

Mathematics Subject Classification: 97K20, 90C59

1. Introduction

The Vehicle Routing Problem (VRP) is a generic name given to a whole class of problems in which a set of routes for a fleet of vehicles based at one or several depots must be determined for a number of geographically dispersed cities or customers, see Figure 1. The objective of the VRP is to deliver a set of customers with known demands on minimum-cost vehicle routes originating and terminating at a depot. The interest in VRP is motivated by its practical relevance as well as by its considerable difficulty. The VRP arises naturally as a central problem in the fields of transportation, distribution, and logistics. In some market sectors, transportation means a high percentage of the value added to goods. Therefore, the utilization of computerized methods for transportation often results in significant savings ranging from 5% to 20% of the total costs, as reported in [15]. Usually, in real world VRPs, many side constraints appear. Some of the most important restrictions are:

- Every vehicle has a limited capacity (Capacitated VRP - CVRP)
- Every customer has to be supplied within a certain time window (VRP with time windows - VRPTW)
- The vendor uses many depots to supply the customers (Multiple Depot VRP - MDVRP)
- Customers may return some goods to the depot (VRP with Pick-Up and Delivering - VRPPD)
- Customers may be served by different vehicles (Split Delivery VRP - SDVRP)
- Some values (like number of customers, theirs demands, serve time or travel time) are random (Stochastic VRP - SVRP)

---

[1] Corresponding author. E-mail: martin.kotyrba@osu.cz

- The deliveries may be done in some days (Periodic VRP - PVRP)

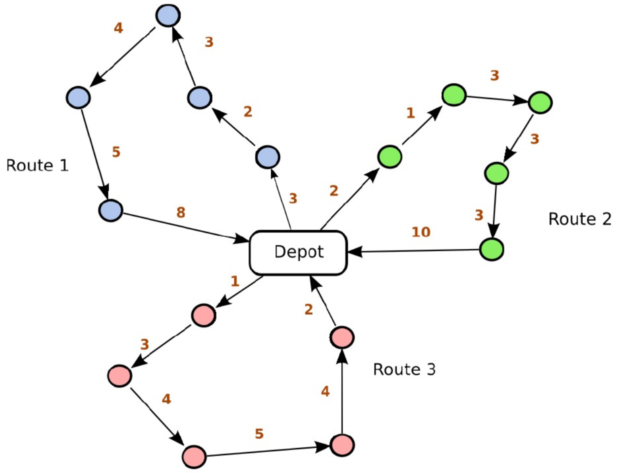


Figure 1: The vehicle routing problem (adapted from http://neo.lcc.uma.es/dynamic/vrp.html)

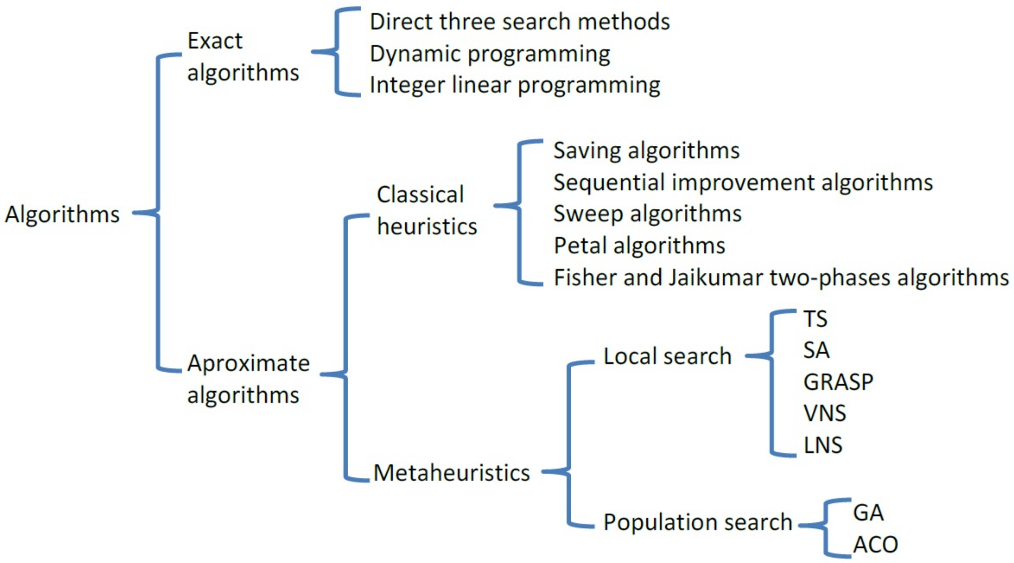


Figure 2: The algorithms for VRP and their relation, adapted from [11]

The Vehicle Routing Problem (VRP) is NP-Hard, and therefore difficult to solve. The fact that VRP is both of theoretical and practical interest (due to its real world applications) explains the amount of attention given to the VRP by researchers in the past years. The vehicle routing problem has been very extensively studied in the optimization literature. It started with the seminal papers of [3]. The literature of VRP is classified into *exact methods*, *heuristics approaches*, *meta-heuristics*, and *hybrid methods*. Figure 2 summarizes the relation between the algorithms for VRP. We refer to [10] for exact, heuristic and metaheuristic algorithms, and to [1] for recent exact algorithms applied to the VRP. Exact algorithms can only tackle problems of a relatively small scale. Approximate algorithms are able to find very near-optimal solutions for large-scale problems within a very satisfactory computation time, and thus are commonly used in practice. A variety of approximate algorithms, including classical heuristics and metaheuristics since the 1980s, are proposed in the literature to efficiently solve different variants of VRP. Most of the models developed for the VRP in the literature considered deterministic parameters, such as deterministic travel times, demands, and service times. Compared with the classical heuristics, metaheuristics carry out a more thorough search of the solution space, allowing inferior and sometimes infeasible moves, in addition to re-combining solutions to create new ones. As a result, metaheuristics are capable of consistently producing high-quality solutions, in spite of greater computation time than early heuristics [4]. Metaheuristics can be categorized into two main types:

1. *Local search.* Local search based methods keep exploring the solution space by iteratively moving from the current solution to another promising solution in its neighborhood. The main local search based metaheuristics for VRP include tabu search (TS) [1], simulated annealing (SA) [2], Greedy Randomized Adaptive Search Procedure (GRASP) [9], Variable Neighborhood Search (VNS), and Large Neighborhood Search (LNS) [13].
2. *Population search.* Population search based methods maintain a pool of good parent solutions, by continually selecting parent solutions to produce promising offspring so as to update the pool. Typical examples are Genetic Algorithms (GA) [19] and Ant Colony Optimization (ACO) [6].

Hybrid methods use a combination of exact, heuristic procedure or meta-heuristics to solve the VRP. Under hybrid methods, a limited number of researches which combine the meta-heuristics with exact methods are presented. Hybrid approaches to evolutionary computation (EC) combine their power with the use of specific heuristics [16] or [17]. The literature reveals that very few researches have applied hybrid methods to vehicle routing problem. Hence, future researchers may focus on developing efficient hybrid approaches by combining two or more of the exact methods, heuristics, and metaheuristics.

2. Mathematical background of VRP

The vehicle routing problem (VRP) is a difficult combinatorial problem, which conceptually lies at the intersection of these two well-studied problems:

- The Traveling Salesman Problem (TSP) is the problem of a salesman who wants to find, starting from his hometown, the shortest possible trip through a given set of customer cities and to return to its hometown; visiting each city only once. TSP can be represented by a complete weighted graph $G = (V, E)$ with V being the set of nodes, representing the cities, and E the set of edges fully connecting the nodes V. The arc set $T \subseteq A$ is a solution for the TSP if it is a simple cycle of length $|V|$ in G. Each edge is assigned a value $d_{ij,}$ which is the length of edge $(i, j) \in A$ that is, the distance between cities i and $j,$ with $i, j \in N$. The TSP is the problem of finding a minimal length Hamiltonian cycle of the graph.
- The Bin Packing Problem (BPP) consists of packing a set of items into a number of bins so that the total weight, volume, etc. does not exceed a maximum value. Mathematically the problem's formulation can be as follows: Given a finite set of elements $E = \{e_1,...,e_n\}$ with associated weights $W = \{w_1,...,w_n\}$ such that $0 \le w_i \le w(bin)$. Partition E into N subsets such that the sum of weights in each partition is at most $w(bin)$ and that N is the minimum.

Hence, we can think of the first transformation as the underlying packing (BPP) structure and the second transformation as the underlying routing (TSP) structure. A feasible solution to the full problem is a TSP tour that also satisfies the packing constraints. Because of the interplay between the two underlying models (both of them are NP Hard problems), instances of the Vehicle Routing Problem (VRP) can be extremely difficult to solve in practice.

The VRP is a combinatorial problem whose ground set is the edges of a graph $G\ (V,E)$. The notation used for this problem is as follows:

- $V = \{v_0, v_1, \ldots, v_n\}$ is a vertex set, where we presume a depot to be located at v_0.
- Let $V' = V \setminus \{v_0\}$ be used as the set of n cities.
- $A = \{(v_i, v_j) \mid v_i, v_j \in V;\ i \neq j\}$ is an arc set.
- C is a matrix of non-negative costs or distances c_{ij} between customers v_i and v_j.
- d is a vector of the customer demands.
- R_i is the route for vehicle i
- m is the number or vehicles (all identical). One route is assigned to each vehicle.

When $c_{ij} = c_{ji}$ for all $(v_i, v_j) \in A$ the problem is said to be symmetric and it is then common to replace A with the edge set $E = \{(v_i, v_j) \mid v_i, v_j \in V;\ i<j\}$.

Each vertex v_i in V' is associated with a quantity q_i of some goods to be delivered by a vehicle. The VRP thus consists of determining a set of m vehicle routes of minimal total cost, starting and ending at a depot so that every vertex in V' is visited exactly once by one vehicle. Making the computation easier, it can be defined (1):

$$b(V) = \left\lfloor \left(\sum_{v_i \in V} d_i \right) / C \right\rfloor \qquad (1)$$

an obvious lower bound on the number of trucks needed to service the customers in set V.

We will consider a service time δ_i (time needed to unload all goods), required by a vehicle to unload the quantity at q_i at v_i. It is required that the total duration of any vehicle route (travel plus service times) may not surpass a given bound D, so, in this context, the cost c_{ij} is taken to be the travel time between the cities. The VRP defined above is NP-hard [17]. A feasible solution is composed of:

- partition $R_1, \ldots, R_m$ of V;
- permutation σ_I of $R_i \cup 0$ specifying the order of the customers on route i.

The cost of a given route ($R_i = \{v_0, v_1, \ldots, v_{m+1}\}$), where $v_i \in V$ and $v_0 = v_{m+1} = 0$ (0 denotes the depot), is given by (2):

$$c(R_i) = \sum_{i=0}^{m} c_{i,i+1} + \sum_{i=1}^{m} \delta_i \tag{2}$$

A route R_i is feasible if the vehicle stops exactly once at each customer and the total duration of the route does not exceed the specified bound D: $c\ (R_i) \leq D$. Finally, the cost of the problem solution S is (3):

$$F_{VRP}(S) = \sum_{i=0}^{m} c(R_i) \tag{3}$$

In this article, we have performed an experimental study that indicates the suitable use of unconventional approaches for the vehicle routing problem.

3. Instances for experimental study

During our experimental study, we focused on Capacitated VRP (CPRV), which was described as follows. CVRP is a Vehicle Routing Problem (VRP) in which a fixed fleet of delivery vehicles of uniform capacity must service known customer demands for a single commodity from a common depot at minimum transport costs. That is, CVRP is like VRP with the additional constraint that every vehicle must have a uniform capacity of a single commodity. A formal description for the CVRP is the following:

- *Objective*: The objective is to minimize the vehicle fleet and the sum of the travel time, and the total demand of commodities for each route may not exceed the capacity of the vehicle which serves that route.
- *Feasibility*: A solution is feasible if the total quantity assigned to each route does not exceed the capacity of the vehicle which services the route.
- *Formulation*: Let Q denote the capacity of a vehicle. Mathematically, a solution for the CVRP is the same that VRP's one, but with the additional restriction that the total demand of all customers supplied on a route R_i does not exceed the vehicle capacity $Q : \sum_{i=1}^{m} d_i \leq Q$.

We tested instances from Capacitated Vehicle Routing Problem Library (CVRPLIB) series A, B, E, M and X, available at [5]. The chosen instances cover different problem sizes (number of cities is 20, 50, 100 and 200) and different distributions of cities (uniform, with a depot on one side and cluster placement), see Table 1. Optimal solutions of all used instances are shown in Table 2.

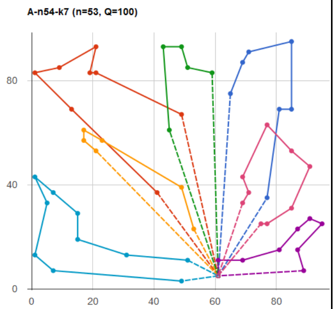
Table 1: The test instances from Capacitated Vehicle Routing Problem Library

Instance	Number of Customers (n)	Minimum Number of Vehicles (K)	Capacity (Q)	Optimal solution
A-n54-k7	53	7	100	1167
B-n50-k7	49	7	100	741
E-n101-k14	100	14	112	1067
E-n22-k4	21	4	6000	345
M-n101-k10	100	10	200	820
M-n200-k16	199	16	200	1274
P-n20-k2	19	2	160	216
P-n50-k7	49	7	150	554

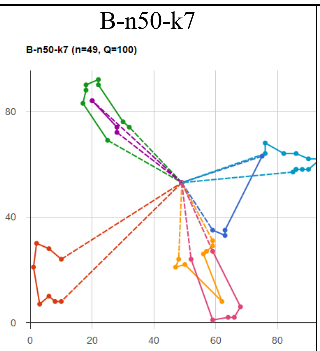
X-n106-k14	105	14	600	26362
X-n200-k36	199	36	402	58578
X-n209-k16	208	16	101	30656

Table 2: Optimal solutions of all used instances

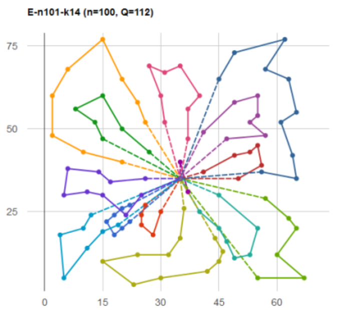
A-n54-k7



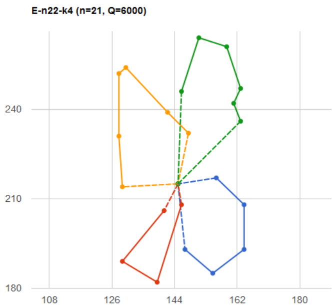
B-n50-k7



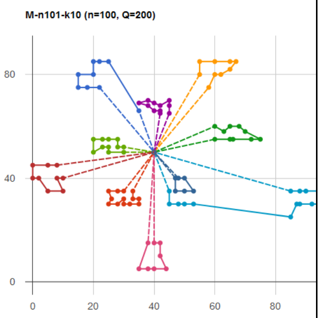
E-n101-k14



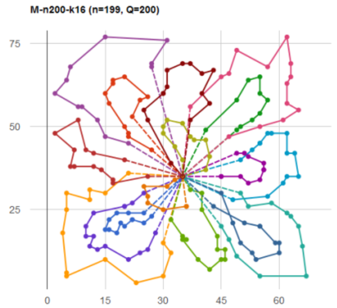
E-n22-k4



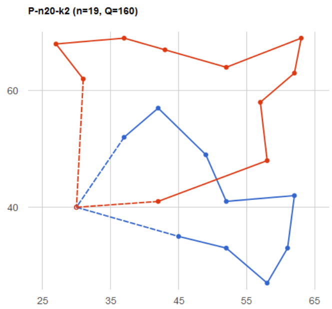
M-n101-k10



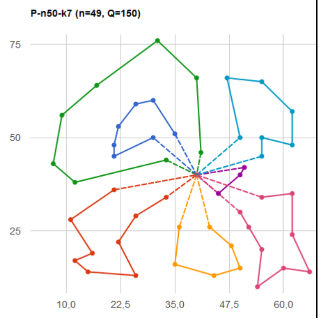
M-n200-k16



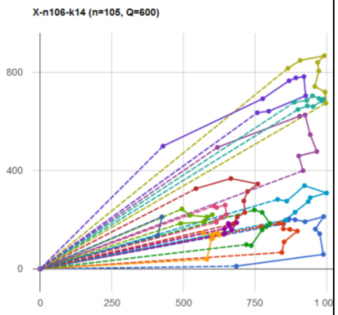
P-n20-k2



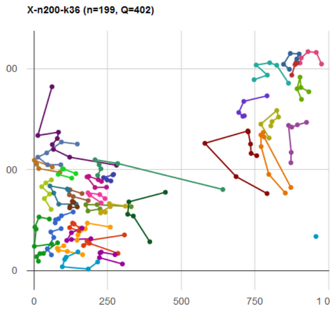
P-n50-k7



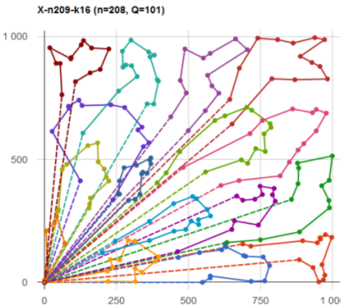
X-n106-k14



X-n200-k36



X-n209-k16



4. Selected heuristics for CVRP

The Clarke and Wright heuristic

The Clarke and Wright (CW) heuristic [3] is one of the best-known and has remained widely practiced to this day despite some of its shortcomings. It is based on the notion of saving. Initially, a feasible solution consists of n back and forth routes between the depot and a customer. At any given iteration, two routes $(v_0, ... v_i, v_0)$ and $(v_0, v_j, .. ,v_0)$ are merged into a single route $(v_0, ... , v_i, v_j, ..., v_0)$ whenever this is feasible, thus generating a saving $s_{ij} = c_{i0} + c_{0j} - c_{ij}$. In the parallel version of the algorithm (CWP), the merge yielding the largest saving is always implemented, whereas the sequential version (CWS) keeps expanding the same route until this is no longer feasible. In practice, the parallel version is much better. Flowcharts of both algorithms are shown in Figure 3.

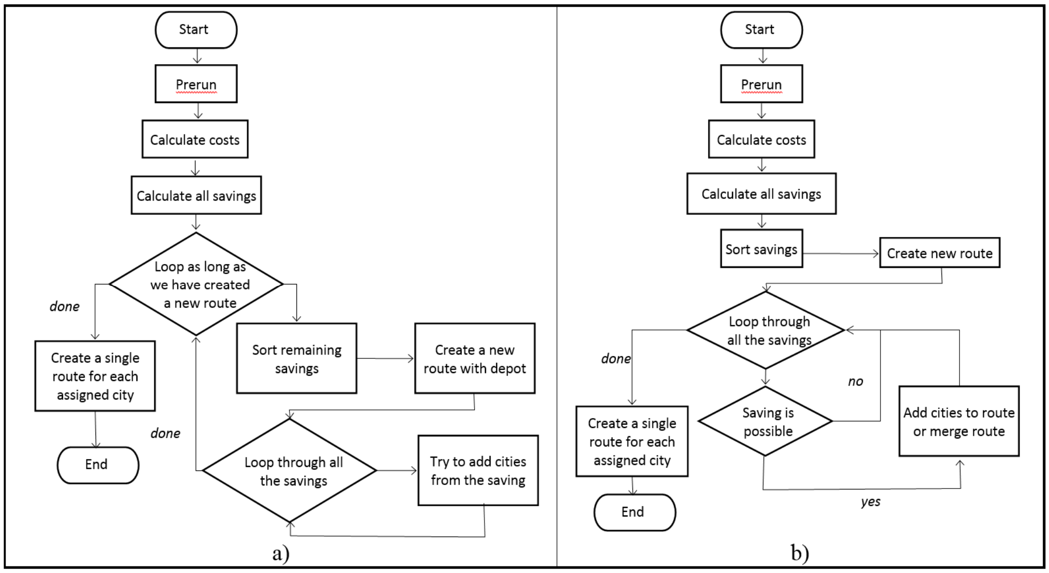

Start
Prerun
Calculate costs
Calculate all savings
Loop as long as we have created a new route
done
Create a single route for each assigned city
End
done
Sort remaining savings
Create a new route with depot
Loop through all the savings
Try to add cities from the saving
a)
Start
Prerun
Calculate costs
Calculate all savings
Sort savings
Create new route
Loop through all the savings
done
no
Saving is possible
Add cities to route or merge route
Create a single route for each assigned city
End
yes
b)

Figure 3: The Clarke and Wright heuristic. a) sequential version (CWS). b) parallel version (CWP).

The sweep algorithm

The sweep algorithm is generally attributed to [7]. It applies to planar instances of the VRP. Feasible routes are created by rotating a ray centered at the depot and gradually including customers in a vehicle route until the capacity or route length constraint is attained. A new route is then initiated and the process is repeated until the entire plane has been swept. An optimization step relating to the created route is then typically applied. This algorithm is also used in the Taillard algorithm. Flowchart of the sweep algorithm is shown in Figure 4a.

The Taillard tabu search algorithm

The Taillard tabu search algorithm [14] uses random tabu durations and continuous diversification. Periodic route reoptimizations are performed by means of an exact TSP algorithm. To help speed up computations, Taillard partitions the problem into several subproblems, each of which is solved independently on a parallel processor. In the case of planar problems, the decomposition process uses concentric rings carved into sectors centered at the depot. For non-planar problems, a different decomposition method based on computation of shortest spanning arborescence is used. The boundaries of the subproblems are redefined dynamically. Flowchart of the Taillard tabu search algorithm is shown in [12].

Genetic algorithms

Genetic algorithms (GA) are a group of methods which may be used to solve search and optimization problems. The basics of GA were laid in [8]. For solving VRP with GAs, it is usual to represent each individual by just one chromosome, which is a chain of integers, each of them representing a customer or a vehicle. So that each vehicle identifier represents in the chromosome a separator between two different routes and a string of customer identifiers represents the sequence of deliveries that must cover a vehicle during its route. Each route begins and end at the depot (it will be assigned the number 0). If we find in the solution two vehicle identifiers not separated by any customer identifier, we will understand that the route is empty and, therefore, it will not be necessary to use all the vehicles available. A typical fitness function used for solving VRP with GA is (4)

$$f_{eval}(x) = f_{\max} - f(x) \tag{4}$$

where

$$f(x) = totaldistance\,(x) + \lambda \cdot overcapacity\,(x) + \mu \cdot overtime\,(x) \tag{5}$$

Both *overcapacity* and *overtime* functions return the amount of capacity and time over the maximum allowed value. If none of the function restriction is violated, *f* returns the total distance travelled. Otherwise, both capacity and time are weighted with values λ and μ. The best solutions may have values close to f_{max}, while the solutions that break any restriction will see penalized their fitness value. Reproduction requires choosing suitable parents first. It can be done quasi-randomly by means of Roulette Wheel on the unit circle [18]. Individuals with the better cost value are more likely to be selected than individuals with the big cost value. Selected parents go to the "mating pool". In the mating pool, two parents are randomly chosen to produce two offsprings. Their chromosomes are recombined by means of crossover and mutation. Crossover means cutting of chromosome in a randomly chosen position and changing parts between parents. After crossover, each offspring goes to the process of mutation. The crossover of *P1* and *P2* works as follows, two cutting sites *i* and *j* are randomly selected in *P1*. Then, the substring *P1*(*i*)···*P1*(*j*) is copied into *C1*(*i*)···*C1*(*j*). Finally, *P2* is swept circularly from *j*+1 onward to complete *C1* with the missing nodes. *C1* is also filled circularly from *j*+1. The other child *C2* may be obtained by exchanging the roles of *P1* and *P2*.

The use of crossover operator is shown in the following example: Let us have two chromosomes *P1* = (9,8,7,5,10,3,6,2,1,4) and *P2* = (9,8,7,6,5,4,3,2,10,1). We choose the substring 5,10,3 that is copied into *C1*. Then *C1* = (−,−,−,5,10,3,−,−,−,−). *P2* is swept circularly from *j*+1 onward to complete *C1* with the missing nodes and also the already used values are omitted. Then *C1* = (−,−,−,5,10,3,2,1,9,8). Now, we continue from the beginning (from the first position). Then *C1* = (7,6,4,5,10,3,2,1,9,8). *C2* is obtained by exchanging the roles of *P1* and *P2*. Therefore, C2 = (7,10,3,6,5,4,2,1,9,8).

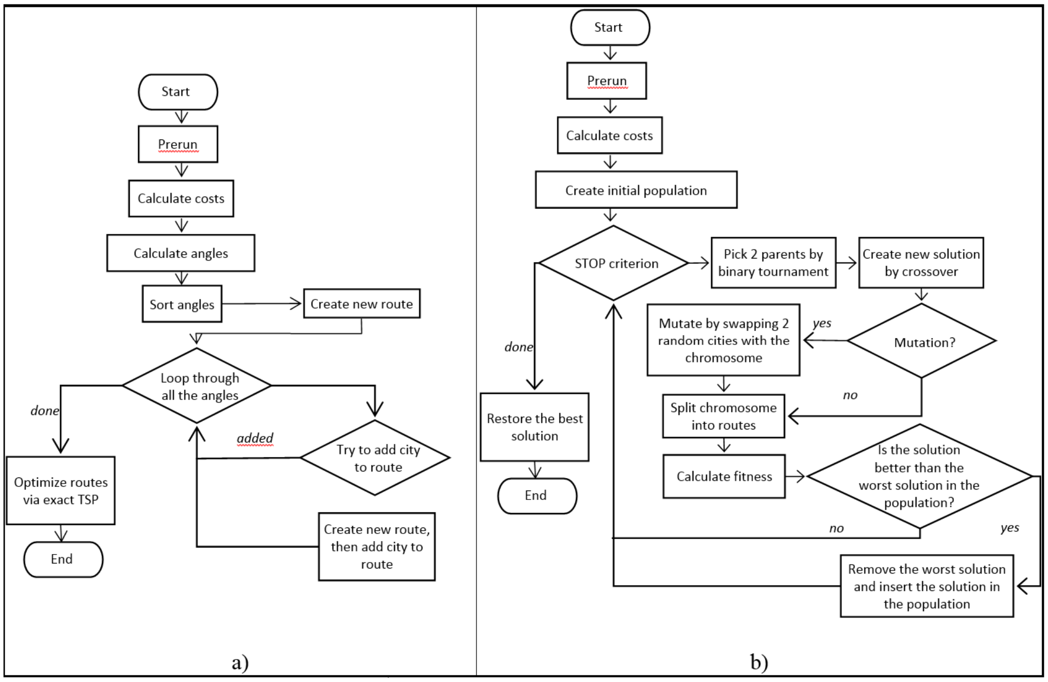

Start
Prerun
Calculate costs
Calculate angles
Sort angles
Create new route
Loop through all the angles
done
added
Try to add city to route
Create new route, then add city to route
Optimize routes via exact TSP
End
a)
Start
Prerun
Calculate costs
Create initial population
STOP criterion
Pick 2 parents by binary tournament
Create new solution by crossover
Mutation?
yes
Mutate by swapping 2 random cities with the chromosome
no
Split chromosome into routes
Calculate fitness
Is the solution better than the worst solution in the population?
no
yes
Remove the worst solution and insert the solution in the population
done
Restore the best solution
End
b)

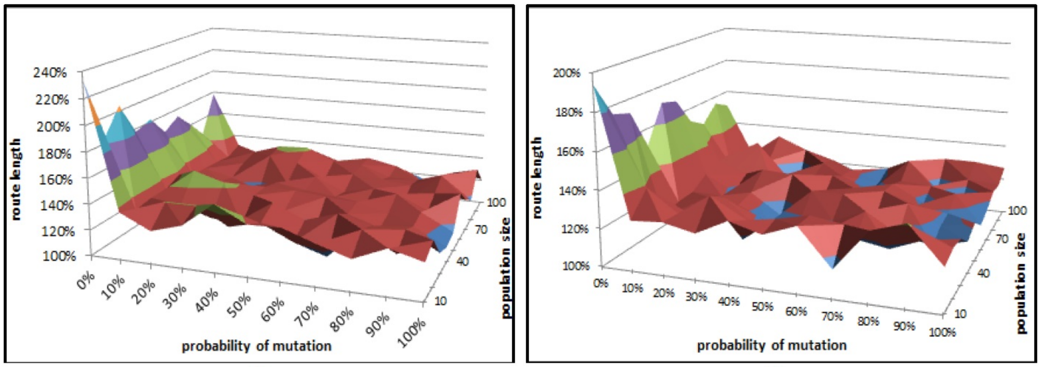
Figure 4: a) Sweep algorithm. b) Genetic algorithms (GA).

The mutation implements an exchange of two randomly selected positions in the chromosome. A flowchart of the used genetic algorithm is shown in Figure 4b.

5. Experimental results

As far as GA, we tested instance A-n54-k7 and P-n50-k7 concerning two parameters *probability of mutation* and *population size* of the four adjustable parameters. We also found a limit regarding the parameter *population size;* after its overcoming the solution does not change. It does not make sense to use a population of less than 10 or greater than 100. We wanted to test the parameter *probability of mutation* in its entire range (0-100%). The calculation was running for all combinations of both parameters population size and mutation probability, and we recorded the quality of all solutions. The *size of population* was incremented in steps of 10 (10, 20, ..., 100) and the *probability of mutation* was incremented in steps of 10% (0%, 10%, ..., 100%). The remaining parameters were assigned with the following values: *maximum number of generations*=1000000, and *maximum number of generations without improvement* =200000. It was found experimentally [12] that solutions practically do not improve after more than 1000000 generations. Another requirement was that the duration of a single run was not extremely long because the number of the tested combinations of the remaining parameters is large.

Furthermore, we found out that if more than 200000 generations were running without improvement solutions, the algorithm is trapped in a local minimum, and there is only a small chance that it will be improved later. Thus, the algorithm usually reached the maximum number of generations without finding any other improvements. Therefore, we limited the *maximum number of generations without improvements* at 200000. We set $\lambda = 1$ and $\mu = 0$ from eq. (6). The obtained experimental results are shown in Figure 5, where *route length* is represented by the ratio of the obtained length of the route towards the optimal solution.



a) b)

Figure 5: The obtained route length is expressed in % compared with the best-known result. a) Tested instance A-n54-k7. b) Tested instance P-n50-k7.

The results of all experiments are shown in Figures 6-8, where the obtained route lengths are expressed in % in comparison with the best-known result. The testing was performed by running all of the above-mentioned algorithms on all tested instances. Each of them was run three times and the best result was recorded. Regarding Clarke and Wright heuristics and the sweep algorithm, there are not parameters that would need to be set. Concerning Taillard's algorithm, the maximum number of iterations without improvement was limited to 10000. As in previous experiments, we limited the maximum number of generations to 1000000, and the maximum number of generations without improvement to 200000 in the case of a genetic algorithm. Then we used the best measured parameters from the previous experiment, i.e. the size of the population that was 85 and the probability of mutation was 90% (the best results were 90 individuals in the population and the probability of mutation 90% for the first experiment, and 80 individuals in the population and the probability of mutation 90% for the second experiment).

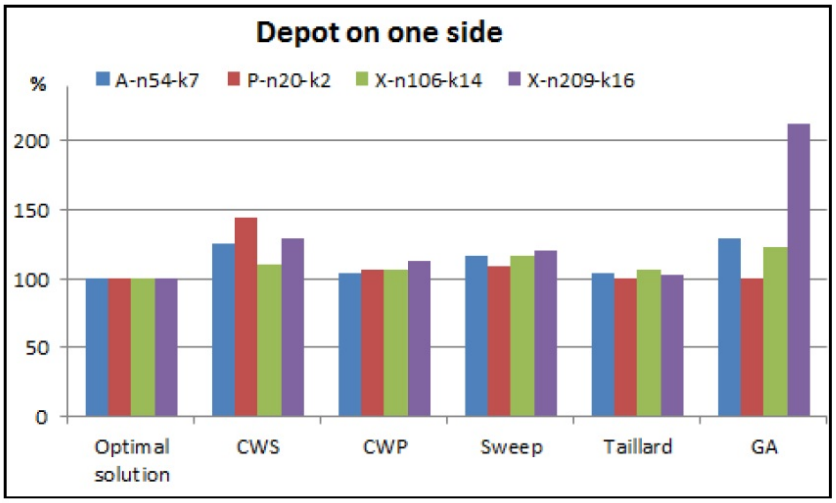


Figure 6: Experimental results of tasks Depot on one side. The ratio of the obtained length of the route is expressed in % in comparison with the best-known result.

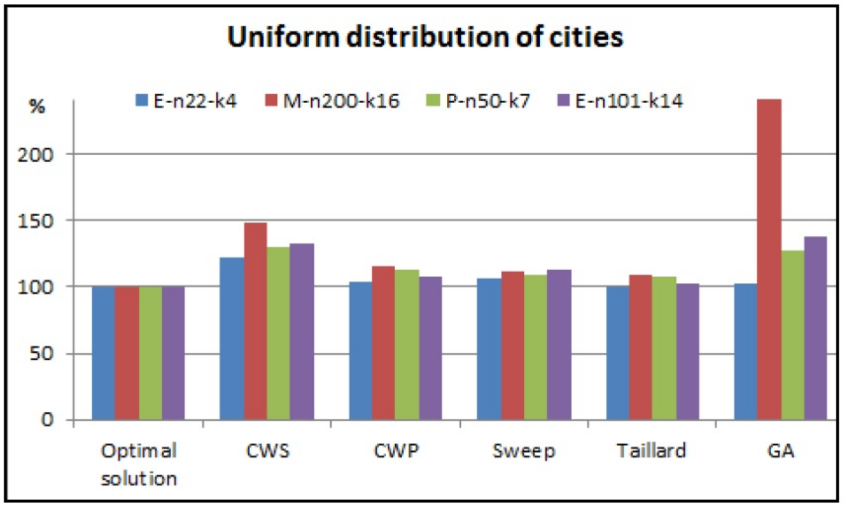


Figure 7: Experimental results of tasks Uniform distribution of cities. The ratio of the obtained length of the route is expressed in % in comparison with the best-known result.

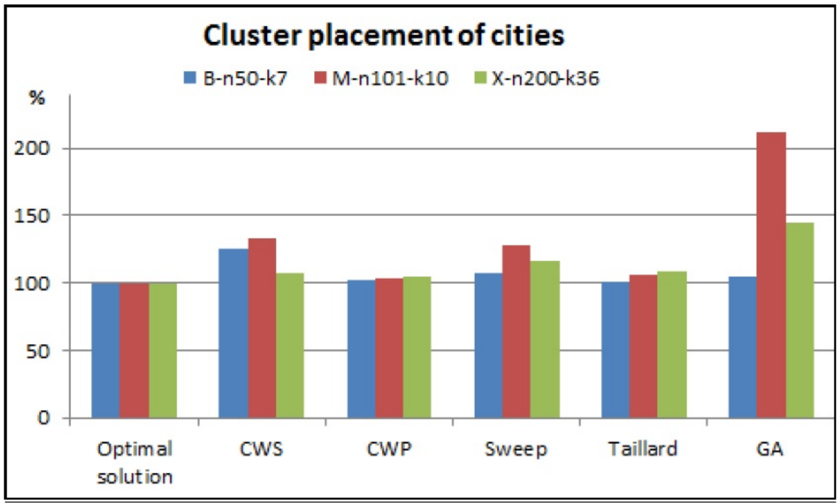


Figure 8: Experimental results of tasks Cluster placement of cities. The ratio of the obtained length of the route is expressed in % in comparison with the best-known result.

6. Conclusion

The Clarke and Wright heuristic scores very high on simplicity and speed. It contains no parameters and it is easy to code. The parallel version of Clarke and Wright heuristics surpassed the sweep algorithm in all instances except for tasks with a uniform distribution of cities. Its strength lies in solving tasks with cluster placement of cities, where CW surpassed even Taillard's algorithm in two tasks. Another CWs advantage is its extreme speed, when the calculation of a task with 200 cities ran within one second. The sweep algorithm scores high on simplicity, but does not seem to be superior to CW both in terms of accuracy and speed. It is also rather inflexible. Again, the greedy nature of the sweep mechanism makes it difficult to accommodate extra constraints and the fact that the algorithm assumes a planar structure severely limits its applicability. In particular, the algorithm is not well suited to instances defined in an urban setting with a grid street layout. The sweep algorithm excelled in tasks with a uniform distribution of cities, where it surpassed algorithm CWP, although it is not so fast. The sweep algorithm does not provide good results compared with Taillard's algorithm.

Taillard's algorithm is one of the best available in terms of accuracy. It has identified nine of eleven best results on the used CVRP instances. Moreover, it found an optimal solution in two instances. The algorithm uses standard insertions, but managing the dynamic decomposition process as well as the parallel implementation adds to its complexity. One would expect this algorithm to handle additional side constraints reasonably well because of the combination of insertion and exchange moves used to define neighbouring solutions. A disadvantage is its time demands.

The proposed implementation of a genetic algorithm uses the crossover of chromosomes and their following allocation of into routes using Prins' algorithm. Details of the used crossover operator are described in Chapter 4. The obtained outputs from our experimental study using genetic algorithms were compared with all of the above-mentioned approaches. The results indicate that genetic algorithms can be used to VRP. A genetic algorithm is able to find a solution with a deviation up to 20%, A genetic algorithm is able to find a solution with a deviation up to 20%, which was verified experimentally. The probability of mutation was set in the range of 3-10%. From the fact that the implemented algorithm achieves excellent results with such an extreme probability of mutation, we can conclude that the used method of crossing was not satisfactory, because all the progress toward better solutions was implemented through mutation. We can see in Figure 5 that the algorithm achieves up to twice the path length (compared to the best-known solution) in the case of no mutation. This is true because the mutation is an important part of genetic algorithms and it enables us to get out of local minima. We can also observe significantly poorer outcomes for the smallest population size of 10. We achieved the best solution for the population size of 70-100 individuals and the probability of mutation 80-100%.

Acknowledgments

The research described here has been financially supported by the University of Ostrava grant SGS/PrF/2016. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] R. Baldacci, A. Mingozzi, R. Roberti and R. W. Calvo, An exact algorithm for the two-echelon capacitated vehicle routing problem. Operations Research, 61 (2013), 298–314.

[2] W. C. Chiang and R. Russell, Simulated Annealing Meta-Heuristics for the Vehicle Routing Problem with Time Windows, Annals of Operations Research, vol. 93, no. 1 (1996), 3-27.

[3] G. Clark and J. W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Operations Research, 12 (1964), 568-581.

[4] J. F. Cordeau, G. Gendreau, G. Laporte, J. Y. Potvin and F. Semet, A guide to vehicle routing heuristice. Journal of the Operational Research Society, 53 (2002), 512–522.

[5] CVRPLIB - Capacitated Vehicle Routing Problem Library, 08 06 2015. [Online]. Available: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/.

[6] L. M. Gambardella, E. Taillard and G. Agazzi, MACSVRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows, in: New Ideas in Optimization, pp. 63-76 London, UK, 1999.

[7] B. Gillett, and L. R. Miller, A Heuristic Algorithm for the Vehicle-Dispatch Problem, Operations Research, vol. 22, no. 2 (1974), 340-349.

[8] J. H. Holland, Adaptation in natural and artificial systems. Ann Arbor: The Univ. of Michigan press, 1975.

[9] G. Kontoravdis and J. A Bard, GRASP for the Vehicle Routing Problem with Time Windows, INFORMS Journal of Computing, vol. 7, no. 1 (1995), 10-23.

[10] G. Laporte, What you should know about the vehicle routing problem. Naval Research Logistics, 54 (2007) 811-819.

[11] C. Lin, K. L. Choy, G. T. Ho, S. H. Chung and H. Y. Lam, Survey of green vehicle routing problem: Past and future trends. Expert Systems with Applications, vol. 41, no. 4 (2014), 1118-1138.

[12] V. Lustek, VRP problem and its variants (in Czech). Diploma thesis. University of Ostrava, 2015.

[13] D. Pisinger, and S. Ropke, Large neighborhood search. In Handbook of metaheuristics. Springer US 2010, pp. 399-419.

[14] E. Taillard, Parallel Iterative Search Methods for Vehicle Routing Problems, Networks, vol. 23, no. 8 (1993), 661-673.

[15] P. Toth and D. Vigo, The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia. 2001.

[16] B. Yu and Z. Z. Yang, An ant colony optimization model: the period vehicle routing problem with time windows. Transportation Research Part E: Logistics and Transportation Review, 47 (2011), 166–181.

[17] T. Vidal and T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei, A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research, 60 (2012), 611–624.

[18] E. Volna, Genetic Algorithms for the Vehicle Routing Problem, In: AIP Conference Proc. 2015 (in print).

[19] K. Zhu, A New Genetic Algorithm for VRPTW, Int. Conf. on Artificial Intelligence, Las Vegas, USA, 2000.