Empirical Analysis of Two Different Metaheuristics for Real-World Vehicle Routing Problems

Tonči Carić[1], Juraj Fosin[1], Ante Galić[1], Hrvoje Gold[1], and Andreas Reinholz[2]

[1] Faculty of Transport and Traffic Sciences, University of Zagreb
Vukelićeva 4, HR-10000 Zagreb, Croatia
{Tonci.Caric,Juraj.Fosin,Ante.Galic,Hrvoje.Gold}@fpz.hr
[2] University of Dortmund, D-44221 Dortmund, Germany
Andreas.Reinholz@gmx.de

Abstract. We present two hybrid Metaheuristics, a hybrid Iterated Local Search and a hybrid Simulated Annealing, for solving real-world extensions of the Vehicle Routing Problem with Time Windows. Both hybrid Metaheuristics are based on the same neighborhood generating operators and local search procedures. The initial solutions are obtained by the Coefficient Weighted Distance Time Heuristics with automated parameter tuning. The strategies are compared in an empirical study on four real-world problems. A performance measure is used that also considers multiple restarts of the algorithms.

Key words: Vehicle Routing Problems with Time Windows, Coefficient Weighted Distance Time Heuristics, Iterated Local Search, Simulated Annealing

1 Introduction

The Vehicle Routing Problem (VRP) is defined by the task of finding optimal routes used by a group of vehicles when serving a group of customers. The solution of the problem is a set of routes which all begin and end in the depot, and which suffices the constraint that all the customers are served only once. The objective is to minimize the overall transportation cost. Transportation cost can be improved by reducing the total traveled distance and by reducing the number of the needed vehicles. By adding only capacity constraints to the VRP problem, it is transformed into the most common variation, the Capacitated Vehicle Routing Problem (CVRP). By adding time constraints to the CVRP in the sense that each customer must be served within a customer specific time window, the VRP turns into the well known Vehicle Routing Problem with Time Windows (VRPTW).

The VRPTW is an important NP hard combinatorial optimization problem [1]. It has a wide applicability and has been the subject of extensive research efforts. The solving of real-world VRPTW for delivering or collecting goods

needs distance and travel time information between customers that is based on a road network. Therefore the Euclidean metric that is widely used in scientific community for VRPTW (e.g. in the Solomon's benchmarks problems [2]) has to be substituted by real data from an Geographic Information System. Considering the bidirectional nature of traffic flows on streets and road networks the access to this data is organized as an asymmetric look-up matrix. Especially the existence of one-way streets in the cities makes the usage of asymmetric matrices very important for the optimization of routes in the urban area. Due to the time constraints, an additional matrix containing forecasted travel times between each pair of customers has to be created. The quality of forecasting can have a high impact on the feasibility of solutions which are executed in the real-world.

For solving VRPTW problems, a large variety of algorithms has been proposed. Older methods developed for the VRPTW are described in the survey [1] and [3]. Most of the new methods tested on Solomon's benchmarks are comprised in [4], [5].

Methods that applied the two-phase approach of solving VRPTW are found to be the most successful [6]. During the first phase the constructive heuristic algorithm is used to generate a feasible initial solution. In the second phase an iterative improvement heuristic can be applied to the initial solution. The mechanism for escaping from local optimum is often implemented in the second phase, too.

This paper describes the method of finding the strategy that needs less time to produce a solution of desired quality. The success of strategies is determined by the time needed to reach the quality threshold with some probability. Algorithm implementations proposed in the paper could be improved step by step by refining and adding more complex and powerful elements and procedures [7].

Reaching and escaping local optimum are important steps in the process of finding the global optimum for the Iterated Local Search (ILS) and Simulated Annealing (SA). Both strategies are developed in the same computational environment in order to have fair conditions for comparison. Both strategies use the same way of reaching the local optimum which is local search procedure with single $\lambda(1,0)$ operator for searching the neighborhood [8]. Escaping from local optimum is done by perturbation procedure which is implemented as k-step move in the solution neighborhood. The initial solution and the number of iterations are the same as well. Iteration is defined as one cycle of algorithm's outer loop. The second step is significantly different for each strategy. In the applied ILS the perturbated solution is brought to local optimum using the local search procedure. If the new local optimum is better than the so far global best, then that solution is the starting point for a new perturbation; otherwise, the escaping step is discarded. On the contrary, Simulated Annealing never sets the global best solution as the starting point for a new iteration. Also, the series of perturbations are allowed if acceptance criteria are activated successively so that there is a possibility of accepting an inferior solution.

The main contribution of this paper is an attempt to implement the known strategies in the form of simple algorithms on real cases and to conduct the statis-

tical analysis for finding the best suitable strategy for each considered problem. Also, the Coefficient Weighted Distance Time Heuristics (CWDTH) is a novel construction algorithm which gives feasible initial solution for all the considered problems. The automated parameter tuning implemented in CWDTH algorithm enables better adaptation of algorithm to problems with different spatial and temporal distribution of customers. The remainder of the paper is organized as follows: In Section 2 the initial solution methodology and improvement strategies are described. Computational experiments and results are given in Section 3. Finally, in Section 4 conclusions are drawn.

2 Solution Methodology

2.1 Initialization

In order to solve the VRPTW problems, a constructive heuristic method CWDTH based on the assignment of weights to serving times and distances to the serving places [9] has been developed, Fig. 1.

```
procedure CWDTH()
    for each k[0, 1] in steps of 0.01 do
        s := NewSolution()
        v := FirstVehicle()
        while not Solved()
            c := BestUnservedCustomer(k)
            Move(v, c)
            if CapacityExhausted(v) then
                Move(v, depot)
                v := NextVehicle()
            endif
        endwhile
        remember s if best so far
    next
    return s
end
```

Fig. 1. Coefficient Weighted Distance Time Heuristics algorithm

Coefficient interval [0, 1] traversed in empirically determined steps of 0.01 is used for the construction of 101 potentially different solutions. In each pass the algorithm starts from an empty solution and selects the first vehicle. Until all customers are served, the routes are constructed by moving the selected vehicle from its current position to the next best not yet served customer.

Procedure $BestUnservedCustomer()$ uses coefficient k to put different weight to distance and time constraint while selecting the next customer to serve. The

criteria of customer selection are:

$$f(x) := MIN(k \cdot Distance(v, c) + (1\text{-}k) \cdot LatestTime(c)) \ .$$

Selection of customer c which minimizes function $f(x)$ depends on the sum of its geographic distance to vehicle v multiplied by coefficient k and its upper bound of time window yield by function *LatestTime* multiplied by 1-k. After the capacity of the selected vehicle is exhausted, it is returned to the depot to complete the route and the next available vehicle is selected for routing. The best of all the generated solutions is returned as initial solution for further optimisation.

Such approach improves the capability of solving VRPTW problems that have different time window configurations. In other words, the algorithm uses the automated parameter tuning for better adaptation to the specific problem.

2.2 Local search

The local search does not solve the VRP problem from the start, but rather requires in-advance prepared feasible solution obtained by some other method, e. g. CWDTH. The local search generates the neighborhood of the given solution and thus successfully reduces the number of potential solutions that will be candidates for the next iteration.

The mechanism of generating local changes, which is the basis for the success of the iterative local search, is performed by single relocation of the customer from one route into another over the set of all route pairs [8]. On such a way the neighborhood $N_i(s)$ is generated where s stands for the seed solution. The principle in which the $\lambda(1, 0)$ operator modifies the routes is presented in Fig. 2.

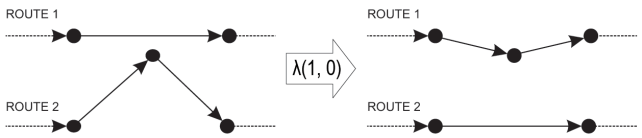

Fig. 2. Local search operator $\lambda(1, 0)$

By iterative procedure this local search tries to improve the solution until it is stuck in the local optimum. In each iteration, from the neighborhood of all the feasible moves that respect time and capacity constraints, the best move that produces the most significant saving is chosen to improve the current solution, Fig. 3.

```
procedure LocalSearch(s)
    terminate := false
    do
        find best candidate solution s' in neighborhood N_i(s) produced by λ(1,0)
        if f(s') <f(s) then
            s := s'
        else
            terminate := true
        endif
    while not terminate
    return s
end
```

Fig. 3. Local search procedure

2.3 Implementation of perturbation

Perturbation operator k-step move uses the same operator $\lambda(1,0)$ as local search procedure but instead of the best move, random move is chosen to modify the solution, Fig. 4.

```
procedure RandomLocalSearch(s)
    choose random candidate solution s' from neighborhood N_i(s) produced by λ(1,0)
    return s'
end
```

Fig. 4. Random local search procedure

This process is k times repeated during one perturbation. The described perturbation gives a feasible solution regarding vehicle capacity and time constraints, Fig. 5.

The number k is generated by binomial distribution generator with success probability p and number of trials n. Values for p and n are empirically obtained. Parameter n is set to number of customers in VRPTW problems and parameter p is set to value $1/n$.

2.4 Iterated Local Search

The local search process is started by selecting an initial candidate solution and then proceeded by iteratively moving from one candidate solution to the neighboring candidate solution, where the decision on each search step is based on limited amount of local information only. In Stochastic Local Search (SLS) algorithms, these decisions as well as the search initialization can be randomized [10].

```
procedure Perturbate(s)
    n := CustomerCount()
    p := 1 / n
    k := BinomialDistribution(p, n)
    for i :=1 to k
        s := RandomLocalSearch(s)
    next
    return s
end
```

Fig. 5. Perturbation procedure

Generally, in the Iterated Local Search (ILS) two types of SLS steps are used [11]. One step for reaching local optima as efficiently as possible and the other for efficiently escaping local optima. Fig. 6 shows an algorithm outline for ILS. From the initial candidate solution provided by CWDTH algorithm, local search procedure is performed. Then, each iteration of ILS algorithm consists of three major stages: first, a perturbation is applied to the current candidate solution *s*. This yields a modified candidate solution *s'* from which in the next stage subsidiary local search procedure is performed until a local optima *s"* is obtained. In the last stage the new global best solution is updated. The algorithm stops after the termination criterion is met.

```
procedure ILS()
    init := CWDTH()
    s := LocalSearch(init)
    best := s
    while not Terminate() do
        s' := Perturbate(s)
        s" := LocalSearch(s')
        if (f(s") <f(best)) then
            best := s"
            s := best
        endif
    endwhile
    return best
end
```

Fig. 6. Iterated Local Search Algorithm

2.5 Simulated Annealing

Simulated Annealing is a stochastic relaxation technique that finds its origin in statistical mechanics [12], [13], [14]. The name of the method comes from

analogy with the annealing process in metallurgy. In the annealing process the material that is heated at high temperature slowly cools and crystallizes under the outside control. Since the heating process allows random movement of atoms, sudden cooling prevents the atoms from achieving the total thermal equilibrium. When the cooling process goes slowly, atoms have enough time to achieve the state of minimal energy forming the ordered crystal grid.

In the optimisation problem the solving of the configuration of atoms is referred to as the state of combinatorial problem, the role of energy is given to cost function and temperature is replaced by control parameter. Simulated Annealing uses stochastic approach to guide the search. The method allows the search to continue in the direction of the neighbor even if the cost function gives inferior results in that direction.

In Simulated Annealing algorithm, the starting solution obtained by CWDTH heuristic and local search procedure is the same as for ILS algorithm and it is set as the global best and as the current solution s as well, Fig. 7.

```
procedure SA()
    T := InitialTemperature()
    init := CWDTH()
    s := LocalSearch(init)
    best := s
    while not Terminate() do
        s' := Perturbate(s)
        s" := LocalSearch(s')
        if (f(s") <f(s)) then
            s := s"
        else
            j := rnd(0, 1)
            k := -((f(s")-f(best))/f(best))/T
            if j <exp(k) then
                s := s"
            endif
        endif
        if (f(s) <f(best)) then
            best := s
        endif
        T := CoolingSchedule()
    endwhile
    return best
end
```

Fig. 7. Simulated Annealing algorithm

At each iteration of SA the k-step perturbation produces solution s'. The perturbated solution s' is additionally improved by the local search producing a new solution $s"$. If a new solution $s"$ is better than the current solution s, it

is accepted as a new current solution. Otherwise, if random generated number within interval [0, 1) is smaller than the current value of acceptance criteria, i.e. $exp(-((f(s")-f(best))/f(best))/T)$, even a worse solution is accepted as the current one. The global best solution is updated if the newly generated solution is better. Initial temperature and cooling schedule are empirically determined once during construction of algorithm and remain the same for all real-world problems and Solomon's benchmark.

2.6 Benchmark results

Before application of algorithms on real-world problems, CWDTH initial solution algorithm and ILS and SA strategies were tested on the standard Solomon's benchmark problems [2]. Comparison of obtained results with the competent results from the literature is shown in Table 1. Testing of both strategies was performed on the 30 independent runs and 4000 iterations as termination criteria.

Table 1. Comparison of results obtained by CWDTH, ILS and SA to the best recently proposed results for Solomon's VRPTW problems. CM stands for cumulative values, CPU stands for the processor characteristics and execution time.

	R1	R2	C1	C2	RC1	RC2	CM	CPU
HG [15]	12.08	2.82	10.00	3.00	11.50	3.25	408	Pentium 400
	1211.67	950.72	828.45	589.96	1395.93	1135.09	57422	3 runs; 1,6 min
BC [16]	12.08	2.73	10.00	3.00	11.50	3.25	407	Pentium 933
	1209.19	963.62	828.38	589.86	11389.22	1143.70	57412	1 run; 512 min
PR [17]	11.92	2.73	10.00	3.00	11.50	3.25	405	Pentium 3000
	1212.39	957.72	828.38	589.86	1387.12	1123.49	57332	10 runs; 2,4mi
M [18]	12.00	2.73	10.00	3.00	11.50	3.25	406	Pentium 800
	1208.18	954.09	828.38	589.86	1387.12	1119.70	56812	1 run; 43,8 min
CWDTH	15.08	3.64	10.44	3.50	14.00	4.25	489	Centrino 2000 duo
	1543.42	1436.86	1004.17	815.22	1797.21	1661.52	77556	100 runs; 0,1 min
ILS	13.67	3.55	10.00	3.25	13.25	4.25	459	Centrino 2000 duo
	1257.79	1022.12	839.47	613.44	1443.52	1192.51	59888	30 runs 7 min
SA	13.08	3.27	10.11	3.25	12.63	3.75	441	Centrino 2000 duo
	1282.22	1053.64	901.37	621.14	1444.15	1239.03	61523	30 runs; 7 min

It is interesting to observe numerical values of cumulative number of vehicles in Table 1. This number include all variations of Solomon's and roughly depict robustness of algorithms.

Comparison of results obtained by very simple CWDTH constructive algorithm and algorithms with advance techniques for optimization, shows that simplicity degrades results approximatively for 20%. Additional local search with one operator $\lambda(1, 0)$ guided by ILS or SA basic strategies can improve solutions for 10% more.

2.7 Performance measure

To compare different algorithms or different parameterized algorithms in an empirical study we were using a performance measure [19] that is motivated by the following question: How often do we have to run an algorithm with a concrete parameter setting so that the resulting solutions are equal or better than a requested quality threshold at a requested accuracy level (i.e. 90%). The lowest number of runs that assures these requests is called multi-start factor (MSF). The MSF multiplied by the average runtime of the fixed parameterized algorithm is the performance measure that has to be minimized.

The estimation of the MSF is based on the fact that the success probability p of being better than the requested threshold quality in one run is Bernoulli distributed. Therefore we can use a parameterized maximum likelihood estimator to determine the success probability p for one run. This implies that the success probability for k runs (in k runs there is at least one successful run) is exactly $1-(1-p)^k$ and that the MSF for reaching a requested accuracy level(AL) can be easily computed using a geometrical distribution with success probability p

$$MSF := min(k \in \mathbf{N} \quad with \quad 1-(1-p)^k \geq AL) \ .$$

When using the intermediate results after each iteration of an algorithm, this procedure can also be used to determine the best combination out of stopping criteria (i.e. maximal number of iterations) and number of restarts for a requested quality threshold and accuracy level.

In this paper we were using the statistical data out of 30 runs for each algorithm and problem instance to estimate the success rates and the average runtimes for all stopping criteria up to 4000 iterations.

The statistical analysis was applied to a series of combinations out of three accuracy levels and two quality thresholds. For the accuracy levels we were using the predefined values 90%, 95%, and 99%. The quality thresholds were chosen out of the data by following procedure: The first quality threshold was defined by the quality value that was reached by the worst out of the 25% best runs after 4000 iterations. The second quality threshold was defined by the quality value that was reached by the worst out of the 10% best runs.

3 Computational Results

3.1 Characteristics of data set

Comparison of ILS and SA strategies was performed on four real-world problems. The objective function for all problems is constructed without penalty because all the operators produce a feasible solution. To force reduction of vehicles in the fleet the objective function is defined as a product of the number of vehicles and the total traveled distance. Problems are classified as VRPTW and described in Table 2.

Table 2. Problem characteristics

Problem	Domain	Customers	Vehicles
VRP1	Drugs delivery	64	7
VRP2	Door-to-door delivery of goods	90	3
VRP3	Delivery of consumer goods	107	14
VRP4	Newspaper delivery	154	6

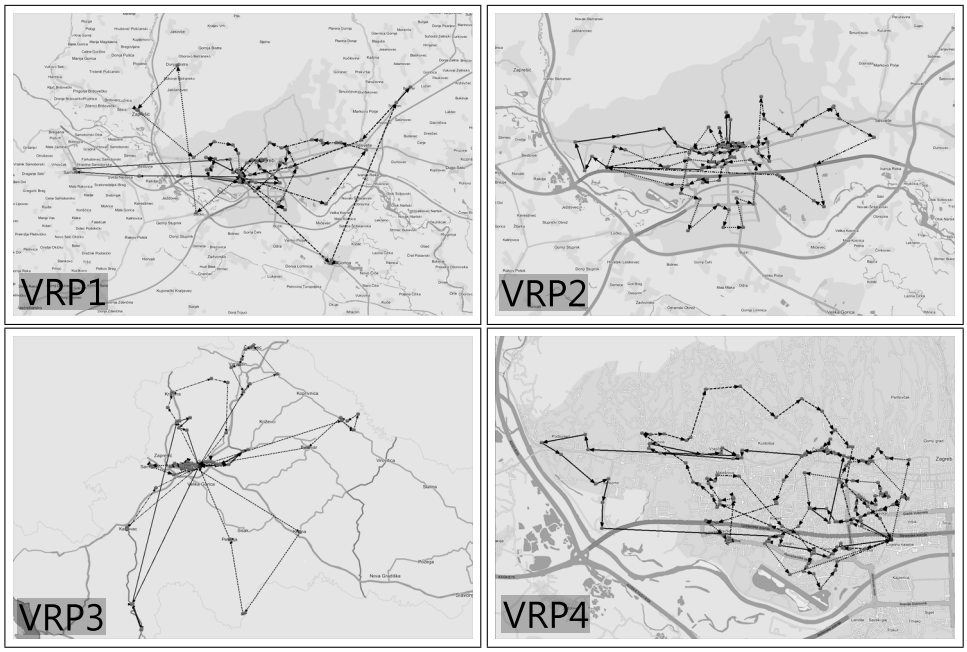

Fig. 8. Geographical distribution of customers for all problems

All problems have homogeneous fleet of vehicles except problem VRP3, and are located in the same geographical area of the city of Zagreb, the capital of Croatia, Fig. 8.

Only one problem, VRP3, spreads on the road networks outside the capital and uses highways between cities. Problems VRP1, VRP3 and VRP4 are mathematically described by distance asymmetric look-up matrix and the related forecasted travel time matrix. The calculation of travel time matrix is based on the average velocity on a particular street or road segments. If such information is not available then the calculation is based on the rank of the road segments. The road ranking follows the national classification which divides them in sixteen categories. Problem VRP2 has linear dependency between the matrix of minimal distances and the matrix of travel times. In problems VRP2 and VRP4 several customers have narrow time windows and in problems VRP1 and VRP3 most of the customers are constrained by terms of working hours of the pickup and delivery department, which means that time windows are relatively wide.

The substitution of the Euclidean metric by the matrix of minimal distances between customers and the use of forecasted travel times for checking time window constraints raises many interesting questions. One of them is the possibility of losing information which can be usable for additional optimisation when we transform real transport networks of streets and roads from geographic information system to asymmetric bidirectional graph with the mentioned asymmetric lookup matrix of minimal distances. At first glance the loss of such information seems to be a problem, but precise analysis leads us to the conclusion that all information that is really important for optimisation of routes are still stored in asymmetric minimal distance matrix. Another important issue is the travel time forecasting model. Such model should be able to predict how much time is needed for a vehicle to move from one geographic location to another in dynamic traffic environment.

3.2 Comparative Analysis

Final results of experiments are shown in Table 3 and Table 4. The examination pool of results was constructed by 240 runs of developed ILS and SA algorithms. In order to determine which strategy needs less time, i.e. number of restarts multiplied by the number of iterations, to produce a solution below the threshold with some accuracy, each problem was solved 60 times (each algorithm 30 runs). Table 3 shows optimal parameters of the winning strategy for all problems.

Parameters from Table 3 guarantee reaching of the threshold interval in minimal time with accuracy of 90%. For example, VRP1 needs to be restarted 17 times with halting criteria set to 1606 iterations per start for ILS algorithm. If we increase the accuracy level the number of restarts increases. For each problem the dependencies of multi-start factor and accuracy level are shown in Table 4.

The thresholds are defined in such a way that all the results obtained by ILS and SA are sorted in a list where the value of objective function on the last iteration is the number on which the sorting is done. Threshold $T1$ is calculated so that 25% of runs in the sorted list are in the $T1$ threshold interval. Threshold

Table 3. Optimal tuning parameters for VRP problems. AL = Accuracy Level, T = Threshold, ALG = Algorithm, MSF = Multi-start factor, IT = Optimal number of iterations per each run

AL	**VRP1**			**VRP2**			**VRP3**			**VRP4**		
90%	*ALG*	*MSF*	*IT*	*ALG*	*MSF*	*IT*	*ALG*	*MSF*	*IT*	*ALG*	*MSF*	*IT*
T1	ILS	9	418	ILS	17	334	SA	9	720	SA	6	1210
T2	ILS	17	1606	ILS	17	3623	SA	34	720	SA	13	1955

Table 4. Multi-start factors for VRP problems. AL = Accuracy Level, T = Threshold, IT = Optimal number of iterations per each run

	VRP1		**VRP2**		**VRP3**		**VRP4**	
AL	*T1* IT=418	*T2* IT=1606	*T1* IT=334	*T2* IT=3623	*T1* IT=720	*T2* IT=720	*T1* IT=1210	*T2* IT=1955
90%	9	17	17	17	9	34	6	13
95%	12	21	21	21	12	44	8	17
99%	26	49	49	49	26	101	18	38

$T2$ has 10% of best runs. Runs that reached the threshold interval $T2$ for all four problems are depicted in Fig. 9.

Statistical analysis of 30 runs gives us the result for VRP1 and VRP2 revealing that ILS will reach the threshold in less time than SA algorithm. It is reasonable to say that ILS algorithm achieved steeper descent of cost function in fewer iterations for VRP1 and VRP2 compared to SA. On the other hand SA algorithm performs better for two larger problems VRP3 and VRP4. The overall best results for problems VRP3 and VRP4 are obtained by SA near the end of the cooling schedule, so that the resulting graphs confirm the expected convergence behavior of SA at very low temperature. The average running times of algorithms for each problem are given in Table 5. All algorithms are coded in programming language MARS [20].

Table 5. Average running time. The CPU time is given for 10 runs and 4000 iteration per run

Problem	**VRP1**		**VRP2**		**VRP3**		**VRP4**	
Algorithm	ILS	SA	ILS	SA	ILS	SA	ILS	SA
CPU [min]	1:03	1:02	4:55	4:46	2:01	1:54	11:31	13:13

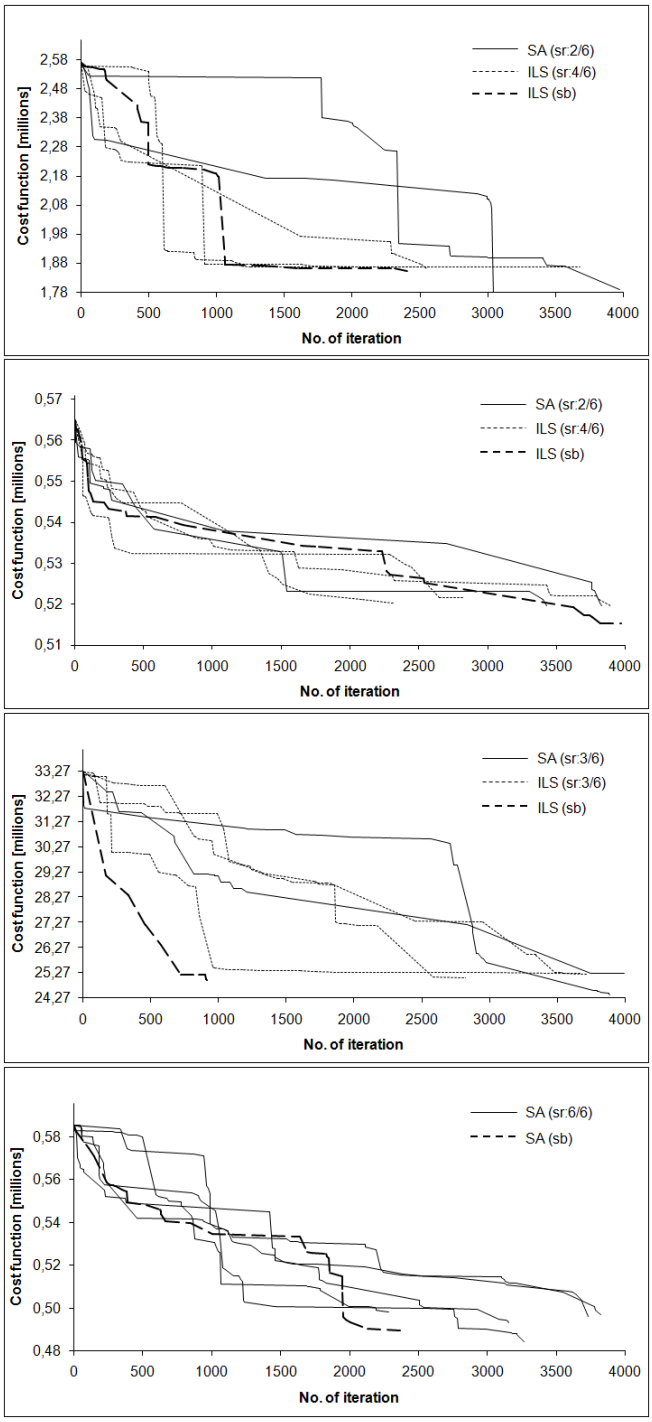

Fig. 9. Results that reached threshold interval $T2$, i.e. 10% best runs, for each VRP problem. SA - Simulated Annealing, ILS - Iterated Local Search, sr - success rate, sb - statistically best

4 Conclusions

Test-bed with four real-world VRPTW problems was set up for comparison of two metaheuristic strategies. The Iterated Local Search and the Simulated Annealing strategies are evaluated in the computationally fair environment using the same procedures such as perturbation and local search, and the same experimental setting like the number of iterations and the initial solution.

A decision criterion for choosing the strategy is optimal time for reaching the threshold interval with the targeted accuracy level. The time is represented as the product of the number of runs and the time for reaching maximal iterations per each run. The threshold interval is defined empirically by the number of best runs. The accuracy level is the probability to reach the threshold in the defined time. The results of the conducted experiments give optimal number of restarts and the number of iterations for each run and also refer to the strategy which is best to use to achieve the threshold in minimal time.

The developed algorithms and the applied statistical procedures show that we can validly choose between the well known ILS and SA strategies for a set of test problems. The increase of accuracy level has as a consequence the increase of the number of runs but does not change the number of iterations for this particular set of examined problems.

There is no obvious lead to state that ILS or SA is better than the other one for all the examined problems, but the results show that ILS is better for smaller problem instances and that ILS reaches threshold interval faster. SA works better for larger problems. The application of the described methodology may be able to help a practitioner to roughly approximate the running time with minor changes in topology and constraints of a problem.

The implemented CWDTH algorithm could be a good choice for the initial feasible solution because of its simple implementation especially in the case of practical problems.

New real-world problems with asymmetric matrices of minimal distances and forecasted travel times are introduced.

References

1. Cordeau, J-F., Desaulniers, G., Desrosiers, J., Solomon, M., and Soumis, F.: The Vehicle Routing Problem with Time Windows. In: Toth P. and Vigo D. (eds.): The Vehicle Routing Problem, SIAM Publishing: Philadelphia, (2002) 157–193
2. Solomon, M.: Algorithms for the Vehicle Routing and Scheduling Problems with Time Windows Constraints. Operations Research. **35** (1987) 254–265
3. Laporte G.: The Vehicle Routing Problem: An Overview of Exact and Approximative Algorithms. European Journal of Operational Research. **59** (1992) 345–358
4. Bräysy, O., Gendreau, M.: Vehicle Routing Problem with Time Windows Part I: Route construction and local search algorithms. Trans. Sci. **39** (2005) 104–118
5. Bräysy, O., Gendreau, M.: Vehicle Routing Problem with Time Windows Part II: Metaheuristics. Trans. Sci. **39** (2005) 119–139

6. Bräysy, O., Dullaert, W.: A Fast Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows. International Journal on Artificial Intelligence Tools. Vol. 12, Issue 2, (Jun 2003) 153–173
7. Reinholz, A.: A Hybrid (1+1)-Evolutionary Algorithm for Periodic and Multiple Depot Vehicle Routing Problems. The 6th Metaheuristics International Conference, Vienna, Austria, August 22-26, 2005, Proceedings on CD. University of Vienna, Department of Business Administration (2005) 793–798
8. Osman I.: Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problems. Annals of Operation Research. **41** (1993) 421–451
9. Galić, A., Carić, T., Fosin, J., Ćavar, I., Gold, H.: Distributed Solving of the VRPTW with Coefficient Weighted Time Distance and Lambda Local Search Heuristics. In: Biljanović, P., Skala, K. (eds.): Proceedings of the 29th International Convention MIPRO, MIPRO, Rijeka (2006) 247–252
10. Hoos, H., Sttzle, T.: Stochastic Local Search: Foundation and Application. Morgan Kauffman, San Francisco (2005)
11. Laurenço, H.R., Serra, D.: Adaptive search heuristics for the generalized assignment problem. Mathware & Soft Computing, **9** (2-3) (2002) 209–234
12. Kirkpatrick, S., Gelatt, C.D. Jr., Vecchi, M.P.: Optimization by Simulated Annealing. Science. **220** (1983) 671–680
13. Cerny, V.: A Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm. Journal of Optimization Theory and Applications. **45** (1985) 41–51
14. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H.: Equations of State Calculations by Fast Computing Machines. The Journals of Chemical Physics. **21** (1953) 1087–1092
15. Homberger, J., Gehring, H.: A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. European Journal of Operational Research. **162** (2005) 220-238
16. Le Bouthillier, A., Crainic, T.G.: Cooperative parallel method for vehicle routing problems with time windows. Computers and Operations Research. **32** (2005) 1685-1708
17. Pisinger, D., Röpke, S.: A general heuristic for vehicle routing problems. Technical Report, Department of Computer Science, University of Copenhagen (2005)
18. Mester, D., Bräysy, O., Dullaert, W.: A multi-parametric evolution strategies algorithm for vehicle routing problems. Expert Systems with Applications. **32** (2007) 508-517
19. Reinholz, A.: Ein statistischer Test zur Leistungsbewertung von iterativen Variationsverfahren. Technical Report 03027, SFB559, University of Dortmund (2003) (in German)
20. Galić, A., Carić, T., Gold, H.: MARS - A Programming Language for Solving Vehicle Routing Problems. In Taniguchi, E., Thompson, R. (eds.): Recent Advances in City Logistics. Proceedings of the 4th International Conference on City Logistics. Elsevier, Amsterdam (2006) 48–57