Success of Heuristics and the Solution Space Representation

Sándor Csiszár

Department of Microelectronics and Technology, Kandó Kálmán Faculty of Electrical Engineering, Budapest Tech
Tavaszmező út 17, H-1084 Budapest, Hungary
csiszar.sandor@kvk.bmf.hu

Abstract: In the study a developed solution for VRP (Vehicle Routing Problem) and its algorithms were analysed in order to find explanations for their success and evaluate their results. Another exciting question at the combinatorial optimization is the solution space which is often referred in the literature despite we do not have tools for its representation, although the distinction of solutions seems to be important at the Evolutionary techniques and at the generalized model of Adaptive Memory Programming.

Keywords: combinatorial optimization, heuristics, vehicle routing problem

1 Introduction

A two-phase metaheuristic with guided route elimination was developed by the author [1] for the Vehicle Routing Problem with Time Windows (VRP TW) to solve a middle size ($n = 100$) combinatorial optimization problem. It is known from experience and from the literature that heuristics are very successful despite their performance is not justified only some convergence theorems are proved [3]. Several questions often arise regarding the

- performance of the heuristics,
- optimality of the found solution,
- navigation in the solution space,
- and how more efficient methods can be found.

It is really difficult to give general answers for these questions (perhaps impossible for one or the other). The major part of the presented results comes from computation and from analysis of a specific problem. The purpose of this article is to better understand and evaluate the performance of the developed algorithms in order to solve large scale problems -where the number of customers is about 1000 and to put these questions into focus.

Regarding the topic of this article (detailed above) only a short VRP definition is given. Let $G = \{V, E\}$ be a graph and $V = \{v_0, v_1, \ldots, v_n\}$ a set of vertices representing the customers around a depot, vertex v_0 denotes the depot. A route is starting from and entering the depot: $\{v_0, \ldots, v_i, v_j, \ldots, v_0\}$. The cost of a route is: $C_t = \sum d_{ij}$, where i and j are consecutive customers on the route and $d_{i,j} \in E$. The objectives of the solution are to determine the lowest number of routes (N_v) and the lowest cost or total travel distance, $C = \sum_{(s)} C_t$, where s is the actual solution, provided that each customer can be visited only once and should satisfy all the constraints (time window, capacity).

2 Investigation of Heuristics Based on the Developed Solution

2.1 Evaluation of the Found Best Solutions

The computational program developed for the cited study [1] and the Solomon benchmark set [4] – including 56 problem instances – was used for the investigation. The benchmark set was developed by M. M. Slomon to compare the behaviour of different heuristics on diverse problems. It seems to be useful and informing to compare the found best result and the estimated lower cost limit of the given problem instance. The lower cost limit was computed the following way:

The number of arcs of the graph is $n(n-1) = 9900$ (directed graph). The shortest $2N_v$ of $\{v_0\ v_j\}\quad j \in \{1,2 \cdots 100\}$ arcs were selected because in case of N_v routes there are as many routes starting from and ending at the depot. The further number of arcs ($100 - N_v$) were selected from the remainder, similarly the shortest ones. At this latest selection the identical values were deleted because only one of them can be applied in a feasible solution (if v_{ij} arc is used then v_{ji} is excluded).

Considering the large number of problem instances (56) and the diversive problem types (random: R100 and R200, cluster: C100 and C200, and semiclustered: RC100 and RC200) the found solutions by heuristics are quite close to the estimated lower cost limit (the reference is the random arc selection). The results are illustrated by problem type, because of the slight difference between the individual problems. In Figure 1 the random arc selection, the lower cost limit and the found best result by heuristic are compared. It must be noted that at the random selection the feasibility of the solution was not considered while at the cost limit calculation a ‘static’ feasibility was considered. It means a route

$\{v_i\, v_j\}$ is feasible if $T_{ei} + t_{si} + d_{ij} + t_{sj} \le T_{lj}$ (1)

where t_{si} is the service time at node i, d_{ij} is the distance between i and j, T_{lj} is the latest start of service at node j, T_{ei} is the earliest start of service at node i. Under route construction the condition determined by Equation (1) is usually not sufficient because only a part of the time window is available for travel. Table 1 in the Appendix gives data for all the six problem types.

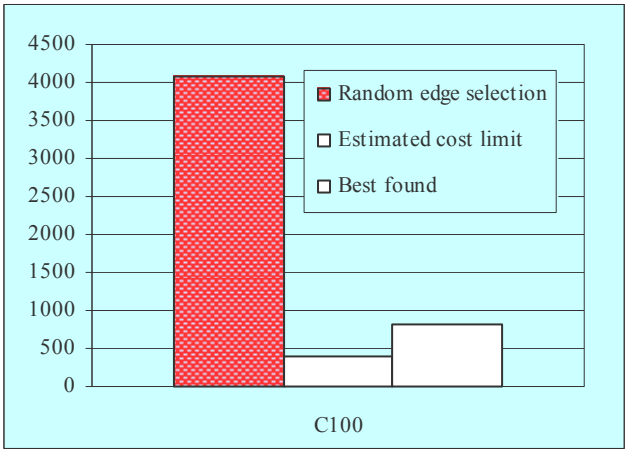

Figure 1
Heuristic performance (clustered problem type)

The investigated heuristics give qiute good results despite they use a so called 'inexact logic'. Explanations for performance are looked for in the next section in case of a specific problem. The following question is investigated in the next section: is it consequential to get good results with heuristics?

2.2 The Nearest Neighbour Example on the Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is a specific case of the VRP where only one route is allowed, consequently the number of solution is: $n!$. Let's examine this problem with the nearest neighbour heuristic (one of the simplest route construction). Let $n = 9$ (number of customers) be in the example (the arc lengths are not essential at this point). Apply the nearest neighbour heuristic for the TSP. Figure 2 shows the route construction. The numbers in the matrix indicate the steps of the route construction procedure. The numbers determine the lowest value within the eligible arcs in each row (the strategy of the nearest neighbour heuristic).

Suppose that an arc of ($v_i - v_j$) is selected. Two things can be established:

1 In every step of the procedure the minimum value of the given row is selected then all the remaining arcs of this row (v_{is}) and the relevant column (v_{rj}) fall out of the further selection ($r, s \in P$, where P is the set of indexes of arcs indicated by 'x' and 'O').

2 If an arc ($v_i - v_j$) is selected –as a minimum value, or travel distance in a certain row- then obviously ($v_j - v_i$) can not be selected.

From the establishments above an inverse-symmetric feature of the matrix is deduced (in the symmetric positions of an 'x' or a number, an 'O' can be found) and the 'x' and numbers in the matrix covers all the arcs of the graph.

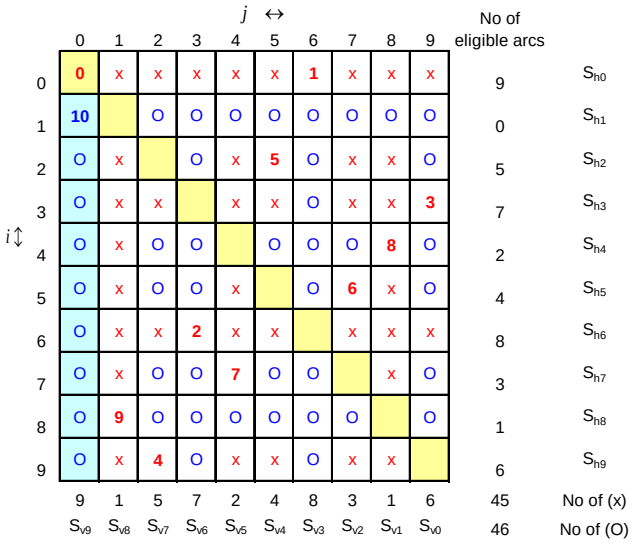

Figure 2
TSP with nearest neighbour route construction

The problem is the following: the last arc (9) is determined by the previous procedure, additionally the last node determines the route (v_{10}) to the starting point (there are two routes to the depot).

Let S_{hi} be the sum of the eligible arcs (d_{ij}) of row i, S_{vj} that of the of the column j. The performance of this simple heuristic can be estimated if 'x' arcs are put in order of magnitude. **Rearrange the rows -and the indexes- of the matrix** according to the decreasing number of 'x':

$$S_{h0} = d_{06} + \sum_{i(j\neq 6)} d_{0j} \text{ , where } d_{06} \le d_{0j}, \qquad j \in \{1,9\} \quad j \neq 6 \tag{2}$$

Generally: $S_{hi} = d_{\min(i)} + \sum_{j \in P} d_{ij}$ (3)

Calculate the average arc: $\overline{a} = \frac{1}{n(n+1)} \sum_{i=0}^{n} \sum_{\substack{j=0 \\ (i \neq j)}}^{n} d_{ij}$ (4)

Let's handle the last arc (d_l) differently. It is known that the numbered arcs (d_{ij}) have the lowest value within the eligible elements. Examine the nearest neighbour heuristic, considering the **arcs signed by 'x' cover all the arcs of the graph:**

$$S_{hi} = k_i \overline{a}_i = k_i d_{\min(i)} + k_i \hat{\Delta}_i \qquad k_i \in \{1, n\} \quad d_{\min(i)} \Rightarrow \hat{\Delta}_i \geq 0 \tag{5}$$

The cost of the solution is:

$$C = \sum_{i=1}^{n} d_{\min(i)} + d_l = \sum_{i=1}^{n} \overline{a}_i - \sum_{i=1}^{n} \hat{\Delta}_i + d_l = n\overline{a} - \sum_{i=1}^{n} \hat{\Delta}_i + \overline{a} + \Delta_l \tag{6}$$

$$C = (n+1)\overline{a} - 0.5 \cdot n(n+1)\ddot{\Delta} + \Delta_l \qquad \ddot{\Delta} \geq 0 \tag{7}$$

$\hat{\Delta}_i$ and $\ddot{\Delta}$ are proportional with the differences of the arcs. If the length of the arcs are identical ($d_{ij} = const$) then the cost of the solution is $(n+1)\overline{a}$, but the found solution is optimal. In any other case the optimal solution is not guaranteed. The higher $\ddot{\Delta}$ (or $\hat{\Delta}_i$) is the closer the found solution by the (nearest neighbour) heuristic to the estimated cost limit. This condition exists at the cluster type problems (C100, C200) where many close and far nodes are available in large number. Although these results were produced by much more sophisticated algorithms but the best numbers of routes at these problems can be easily achieved by initial route construction (Figure 3).

3 Solution Space Representation

At practical problems ($n > 50$) the revealed part of the solution space is very limited. If we take the TSP example, in case of $n = 100$, it has $100! \approx 9{,}33 \cdot \exp(157)$ solutions. Usual computer programs with 3000 iterations using three operators and Global Best Strategy (examining the whole neighbourhood and selecting the best one) investigates maximum 10^8 solutions, that is a negligible part of the solution space. Fortunately the heuristic solutions – according to the experiences – converge very fast to the 'promising zone' and there is no need to search disinterest regions.

Presently we are not able to purposively navigate into the desired zone, but it would be **important to know how far the different solutions are from each other.** The applied search processes don't provide any knowledge about it. It comes from the neighbourhood graph definition [2] that with the available operator set sometimes there is no way from one solution to the other although these solutions are quite 'close' to each other.

An attempt was made in the remaining part of the article to represent the distance of different solutions by their uncommon parts (uncommon arcs). The initial solutions of [1] were used for the exercise. The purpose of this leads to the subject of Adaptive Memory Programming (AMP). AMP is a synthesized methodology. The population of genetic algorithms, the pheromone trail of the ant colony systems or the short- mid - or long term memories of the tabu search play the same role. The basic principle of AMP is to combine components of good solutions in order to construct other, hopefully better solutions. If the new solution is better then the worst one stored in the memory it is updated with the components of the actual one. [5].

In the adaptive memory a kind of crossover is made, consequently the requirement of the solution (which is inserted into the memory) is not only the quality but to have certain differences in order to bring 'new genes' into the next generation of the solutions. **The difference between two solutions is supposed to show the differences in genes.**

Let $0 \le D_{ij} \le 1$ be the distance between two solutions, $E_i, E_j \in E$ the set of arcs of solutions *i* and *j*, then: $D_{ij} = D_{ji} = 1 - \dfrac{card(E_i \cap E_j)}{\min\{card(E_i), card(E_j)\}}$ (8)

The cardinality of arcs in the different solutions is usually the same, VRP is an exception if the number of routes is different. The minimum cardinality is suggested (the range of the received number is wider). If we want to represent more then two solutions then a matrix should be used. In the Appendix a couple of examples are given. In case of several solutions the common part of them is compound, but only the common pairs are calculated.

The calculation of D_{ij} is quite simple, the common part is a scalar product of the relevant matrixes. It is enough to store only those arcs which are in the actual solution (instead of 10 000 Bytes at VRP TW, $n = 100$ only ~120 arcs, 240 Bytes). At the AMP about 10 solutions are stored in the adaptive memory, occupying 2400 Bytes for this purpose.

Tables 2 and 3 show examples. **R203 supposed to have an extent, diverse solution space because with a few exceptions (D_{32}, D_{41}) for most of the other pairs $D_{ij} > 0.7$ relation is valid. C101 seems to have a 'narrow and deep' solution space. It is in accordance with the difficulties of computation** and the

establishments in this article. Coming back to the purpose of distance of solutions –beyond the visualization- it can be used as an indicator at the refreshment of adaptive memory. Although the AMP algorithm ensures a certain diversification, if continuously very 'close solutions' are added to the memory it does not help revealing new solution space.

Conclusions

This study summarises thoughts about the success of heuristics, analyse and explain the received results, raise a few questions related to the solution space, introduces distance of solutions and prepare an Adaptive Memory Programming study with $n = 1000$ customers.

Acknowledgement

The author wishes to thank the Department of Microelectronics and Technology, Budapest Tech, Hungary – first of all Dr. Péter Turmezei, Director of Kandó Kálmán Faculty of Electrical Engineering – for their support.

References

[1] S. Csiszár: Optimization Approaches for Logistic Problems -Vehicle Routing Problem with Time Windows, PhD Dissertation, Budapest, Hungary, submitted: August 2006

[2] I. Futó: Mesterséges Intelligencia (Aula Kiadó), 1999, pp. 256-257

[3] B. Hajek: Cooling Schedules for Optimal Annealing, Mathematics of Operations Research 13, 2001, pp. 311-329

[4] M. M. Solomon: Algorithms for the Vehicle Routing and Scheduling Problems with Time Windows Constraints, Operation Research 35, 1987, pp. 254-265

[5] É. D. Taillard, L. M. Gambadella, M. Gendreau, J. Y. Potvin: Adaptive Memory Programming: A Unified Wiew of Metaheuristics, European Journal of Operation Research 135, 2001, pp. 1-16

Appendix

	Random edge selection	Estimated cost limit	Best found by heuristic
R100	3347.6	532.8	1209.9
R200	3507.7	514.4	951.9
C100	4086.5	405.3	828.4
C200	4214.4	527.5	589.9
RC100	4329.7	519.0	1384.2
RC200	4633.0	521.8	1119.4

Table 1
Heuristic performance (by problem type)

R 203

	1	2	3	4	5	6	7	8	9	10
1	0									
2	0.83	0								
3	0.78	0.35	0							
4	0.36	0.78	0.71	0						
5	0.68	0.81	0.74	0.63	0					
6	0.78	0.83	0.77	0.71	0.29	0				
7	0.90	0.66	0.60	0.85	0.87	0.83	0			
8	0.75	0.84	0.82	0.69	0.80	0.81	0.93	0		
9	0.76	0.56	0.55	0.74	0.77	0.77	0.71	0.86	0	
10	0.66	0.75	0.83	0.52	0.72	0.75	0.85	0.63	0.74	0

Table 2
Solution space representation (R203)

C 101

	1	2	3	4	5	6	7	8	9	10
1	0									
2	0.00	0								
3	0.00	0.00	0							
4	0.00	0.00	0.00	0						
5	0.00	0.00	0.00	0.00	0					
6	0.04	0.04	0.04	0.04	0.04	0				
7	0.04	0.04	0.04	0.04	0.04	0.05	0			
8	0.06	0.06	0.06	0.06	0.06	0.05	0.08	0		
9	0.06	0.06	0.06	0.06	0.06	0.08	0.06	0.10	0	
10	0.06	0.06	0.06	0.06	0.06	0.08	0.06	0.10	0.00	0

Table 2
Solution space representation (C101)

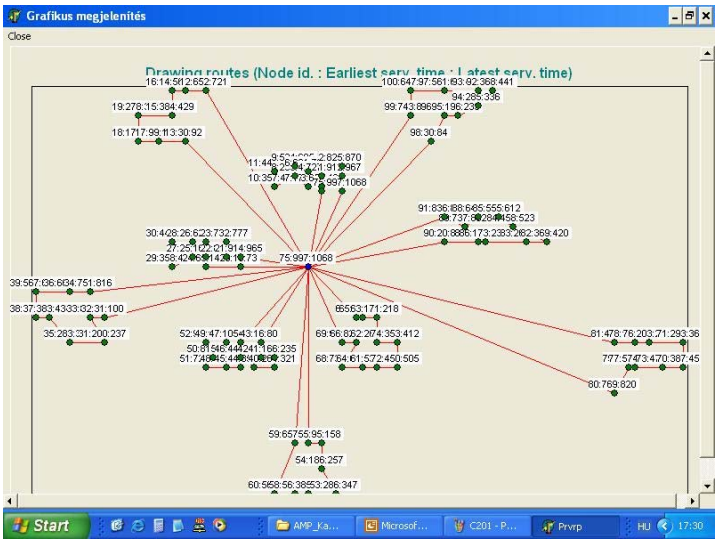

Figure 2
Computational Result of C100 Problem