DOI: 10.15514/ISPRAS-2019-31(3)-12

Constructive heuristics for Capacitated Vehicle Routing Problem: a comparative study

S.M. Avdoshin, ORCID: 0000-0001-8473-8077 <savdoshin@hse.ru>
E.N. Beresneva, ORCID: 0000-0001-6710-2843 <eberesneva@hse.ru>
Department of Software Engineering,
National Research University Higher School of Economics,
20, Myasnitskaya st., Moscow, 101000 Russia

Abstract. Vehicle Routing Problem (VRP) is concerned with the optimal design of routes to be used by a fleet of vehicles to serve a set of customers. In this study we analyze constructive heuristics for a subcase of VRP, where the vehicles have a limited capacity – Capacitated Vehicle Routing Problem (CVRP). The problem is NP-hard, therefore heuristic algorithms which provide near-optimal polynomial-time solutions are still actual. The aim of this work is to make a comparison of constructive heuristics as there were not found any such classification. Finally, the leader by a criterion of quality is admitted being a Clarke and Wright Savings heuristic; however, this algorithm cannot find the solution for all used instances. This fact and other ones are discussed in the paper. Our future goal is to make an experimental comparison of the most common and state-of-the-art metaheuristics using well suited constructive heuristic to build a suboptimal solution.

Keywords: capacitated vehicle routing problem; classical heuristics; constructive heuristics

For citation: Avdoshin S.M., Beresneva E.N. Constructive heuristics for Capacitated Vehicle Routing Problem: a comparative study. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 3, 2019. pp. 145-156. DOI: 10.15514/ISPRAS-2019-31(3)-12

Эвристические методы конструирования маршрута для решения задачи маршрутизации с ограничением по грузоподъемности

С.М. Авдошин, ORCID: 0000-0001-8473-8077 <savdoshin@hse.ru>
Е.Н. Береснева, ORCID: 0000-0001-6710-2843 <eberesneva@hse.ru>
Департамент программной инженерии,
Национальный исследовательский университет "Высшая школа экономики", 101000,
Россия, г. Москва, ул. Мясницкая, д. 20

Аннотация. Задача маршрутизации – одна из широко известных задач комбинаторной оптимизации. Она состоит в отыскании оптимального множества маршрутов для транспортных средств с целью однократного обслуживания определенного множества клиентов. В данной работе исследуется подвид задачи маршрутизации – задача маршрутизации с ограничением по грузоподъемности, в которой каждое транспортное средство имеет свою грузоподъемность. Задача является NP-трудной, поэтому вместо точных алгоритмов решения исследуются только эвристические алгоритмы, позволяющие получить приближенные решения за полиномиальное время. Задача работы – провести экспериментальное исследование точности решения различных конструктивных эвристик, так как в других источниках не было найдено подобных сравнений. В большинстве случаев, лидером можно признать эвристику «Clarke and Wright Savings», однако существуют отдельные наборы данных, описанные в тексте, на которых лучше работают другие алгоритмы. Также в статье рассмотрены и другие интересные факты. В целом работа проделана с целью дальнейшего использования полученных знаний в экспериментальном исследовании наиболее известных и современных метаэвристических

алгоритмов решения задачи маршрутизации с ограничением по грузоподъемности, для которых будут получены предварительные решения на основе выявленных лучших эвристических методов конструирования маршрута.

Ключевые слова: задача маршрутизации с ограничением по грузоподъемности; эвристические методы конструирования маршрута

Для цитирования: Авдошин С.М., Береснева Е.Н. Эвристические методы конструирования маршрута для решения задачи маршрутизации с ограничением по грузоподъемности. Труды ИСП РАН, том 31, вып. 3, 2019 г., стр. 145-156 (на английском языке). DOI: 10.15514/ISPRAS-2019-31(3)-12

1. Introduction

The Vehicle Routing Problem (VRP) is one of the most widely known questions in a class of combinatorial optimization problems. VRP is directly related to Logistics transportation problem and it is meant to be a generalization of the Travelling Salesman Problem (TSP). In contrast to TSP, VRP produces solutions containing some (usually, more than one) looped cycles, which are started and finished at the same point called «depot». The objective is to minimize the cost (time or distance) for all tours. For the identical type of input data, VRP has higher solving complexity than TSP. Both problems belong to the class of NP-hard tasks.

This work is aimed at analysis of VRP subcase, which is called Capacitated Vehicle Routing Problem (Capacitated VRP, CVRP), where the vehicles have a limited capacity. It means that there is a physical restriction on transportation more than determined amount of weight for each machine. Capacitated vehicle routing problems form the core of logistics planning and are hence of great practical and theoretical interest.

There are three types of algorithms that are used to solve any subcase of CVRP.

- *Exact algorithms.* These algorithms find an optimal solution but take a great time for solving large instances. Such methods include Branch-and-Bound, Branch-and-Cut, cutting plane, column generation, cut and solve, Branch-and-Cut-and-Price, Branch-and-Price, and dynamic programming techniques. It was shown in [1] that Branch-and-Bound algorithm was able to solve random CVRP instances with up to 300 customers and four vehicles within 1000 CPU seconds in 2002. However, according to the same source some real-world CVRP instances with up to 47 vertices only were successfully solved within 1000 CPU seconds. Current situation does not differ a lot. State-of-the-art exact methods can provide optimal solution for some SCVRP instances with up to 100 nodes, but it takes 30-40 minutes at average [2]. Due to these restrictions, researchers all over the world concentrate on heuristic methods.
- *Classical heuristics.* These algorithms build an approximate solution iteratively, but they do not include further improvement stage. Different scientific works reveal that, in comparison to exact methods, classical heuristics work much faster. For example, an instance of 100-150 nodes can be solved up to a few (1-2) seconds [2]. Heuristics are divided into two groups that include constructive heuristics and improvement heuristics.
- *Metaheuristics*. Such type of algorithms is also called a framework for building heuristics. According to [3], metaheuristics either explore the solution space by moving at each iteration from a solution to another solution in its neighbourhood (metaheuristics based on local search) or evolve a population of solutions which may be combined together in the hope of generating better ones (metaheuristics based on population, natural inspired).

Actuality of research and development of heuristics algorithms for solving VRP is on its top, because such approximate algorithms can produce near-optimal solutions in a polynomial time. It is especially important in real-world tasks when there are more than one hundred clients in a delivery net. Among the best-known algorithms for CVRP there are metaheuristic proposed by Pisinger and Ropke [4], Nagata and Braysy [5], and Vidal et al [6].

There are a lot of articles related to CVRP heuristics, but no works were found which compare solution quality, or gap, of classical heuristics using the same data bases. Solution quality is calculated as the percentage of difference in the obtained value of the solution with the optimal (or best-known) solution for the problem.

It is important to analyze classical heuristics since constructive heuristics are usually used in order to provide an initial (suboptimal) solution to improvement methods and to metaheuristics that allow to iteratively get near optimal solutions. So, we will discuss only algorithms from the first group.

The paper is structured as follows. In the second part a mathematical formulation of CVRP is given. In the third section, some notes on a classification of most popular constructive heuristics are provided, including description of chosen algorithms. The fourth part consists of design of experiments and their results. And, finally, in the fifth part conclusions and future goals are given.

2. Classical CVRP mathematical model

In the paper we will use CVRP abbreviation having in mind the mathematical formulation that was described in a previous work of authors [7].

Let a complete weighted oriented graph $G = < V, V \times V >$ is given. Let $I = \{0, 1, \dots, N\}$, where $N + 1 = |V|$. Graph vertices are indexed as $ind = V \to I$, $(\forall v \in V)(\forall w \in V)\ v \neq w \Longrightarrow ind(v) \neq ind(w)$. Thus, $V = \{v_0, v_1, \dots, v_N\}$ is the set of vertices, here $i = ind(v_i), i \in I$. Let v_0 be the depot, where vehicles are located, and v_i be the destination points of a delivery, $i \neq 0$.

The distance between two vertices v_i and v_j is calculated using a distance function $c(v_i, v_j)$. Here a real-valued function $c(\cdot,\cdot)$ on $V \times V$ satisfies $\forall i, j, g \in I$ [8]:

- $c(v_i, v_j) \geq 0$ (non-negativity axiom);
- $c(v_i, v_j) = 0$ if and only if $v_i = v_j$ (identity axiom);
- $c(v_i, v_j) = c(v_j, v_i)$ (symmetry axiom);
- $c(v_i, v_g) \leq c(v_i, v_j) + c(v_j, v_g)$ (triangle inequality axiom).

Each destination point v_i, $\forall i \in I$, is associated with a known nonnegative demand, d_i, to be delivered, and the depot has a fictitious demand $d_0 = 0$. The total demand of the set $V' \subseteq V$ is calculated as $d(V') = \sum_{i'=1}^{|V'|} d_{i'}$.

Let K be a number of available vehicles at the depot v_0. Each vehicle has the same capacity – C. Let us assume that every vehicle may perform at most one route and $K \geq K_{min}$, where K_{min} is a minimal number of vehicles needed to serve all the customers due to restriction on C. Clearly, next condition must be fulfilled – $(\forall v_i \in V)\ d_i \leq C, \forall i \in I$, which prohibits goods transportation that exceed maximum vehicle capacity.

Let introduce $V^0 = \{v_0\}$, where $v_0 \in V$. Let us divide V in $K+1$ sets: $S = \{V^0, V^1, \dots, V^K\}$, each subset, except for V^0, represent a set of customers to be served for one vehicle. $S^{all} = \{S\}$ is a set of all possible partitions of V. Let $J = \{0, 1, \dots, K\}$ be a set that keeps indexes. Then $(\forall j \in J)\ |V^j| \geq 1$. There should be no duplicates in any of subsets from S: $(\forall g \in J)(\forall j \in J)(g \neq j \Rightarrow V^g \cap V^j = \emptyset)$. Also, all subsets from S must form set V. Thus, $V = \bigcup_{j=0}^{K} V^j$. In this notation, $V^{0k} = V^0 \cup V^k, \forall k \in J\backslash\{0\}$. It is obvious that $d(V^{0k}) \leq C$.

Let introduce $M^k = \{1, N^1 \dots, N^k\}$, $N^k = |V^k|$, $\sum_{k=1}^{K} N^k = N$. Then let $M^{0k} = \{0\} \cup M^k$. Let $I^k = \bigcup_{k=1}^{K}\{i \mid i = ind(v), \forall v \in V^k\}$ be a set of vertex indices from V^k. Then $I^{0k} = \{0\} \cup I^k, \forall k \in J\backslash\{0\}$.

Let $H^k = \{p^k : M^{0k} \to I^{0k} \mid p^k(0) = 0\ \&\ (\forall x \in M^{0k})(\forall y \in M^{0k})\ x \neq y\ \Longrightarrow p^k(x) \neq p^k(y)\}$ be a set of codes of all possible permutations $h^k = \left(v_{p^k(0)}, v_{p^k(1)}, \dots, v_{p^k(N^k)}\right)$ of V^{0k}. These

permutations represent all possible Hamiltonian cycles of graph $G^{0k} < V^{(0k)}, V^{(0k)} \times V^{(0k)} >$, $\forall k \in J\backslash\{0\}$.

Weight of $h^k \in H^k$ can be found according to the formula 1:

$$f(h^k) = c\left(v_{p^k(0)}, v_{p^k(N^k)}\right) + \sum_{q=0}^{N^k-1} c\left(v_{p^k(q)}, v_{p^k(q+1)}\right) \qquad (1)$$

Let S' be a set of $\{V^{01}, V^{02}, \dots, V^{0K}\}$. In this notation the weight of S' is calculated as $F(S') = \sum_{k=\overline{1..K}} f(h^k), \forall k \in J\backslash\{0\}$.

Overall, the formulation of CVRP is to find:

$$S^0 : F(S^0) = \min_{S \in S^{all}} F(S) \qquad (2)$$

3. Constructive heuristics

In this section the most popular constructive heuristics are described.

3.1 Sequential Insertion algorithm (SI)

Sequential Insertion algorithm [9] constructs routes subsequently, one after another.

In the first step, a new tour $tour_k, k \leq K$, is initialized with a random unrouted node $v_i, i \neq 0$, and the depot v_0. Thus, a tour (v_0, v_i, v_0) is obtained.

In the second step, another unrouted vertex $v_j, j \neq 0$, is chosen, such that its incorporation in the current tour gives the least increase in a tour length and demand of a potential node v_j does not exceed vehicle capacity. So, the next two formulae must be hold:

- $\underset{\substack{v_a \in tour_k, \\ v_{a+1} \in tour_k, \\ v_j \notin tour_k}}{\operatorname{argmin}}\ c(v_a, v_j) + c(v_j, v_{a+1}) - c(v_a, v_{a+1})$;
- $D_{tour_k} + d_j \leq C$, where D_{tour_k} is a total demand of current $tour_k$.

If all conditions hold then this unrouted vertex $v_j, j \neq 0$ is inserted in a tour $tour_k$ between v_a and v_{a+1}.

The second step is repeated until no more unrouted vertex $v_j, j \neq 0$, can be feasibly inserted. In this case a new tour $tour_k, k \leq K$, is initialized, and the procedure starts from the first step.

3.2 Improved Parallel Insertion algorithm

Parallel Insertion Improved algorithm [10] builds routes simultaneously. This method is a modification of Sequential Insertion algorithm.

In the first step, the minimum number K_{min} of feasible routes is defined as $K_{min} = \sum_{i \in |V|} d_i / C$. All these routes $tour_k \in Tours$ are initialized with K_{min} different closest to v_0 unrouted nodes $v_i, i \neq 0$. Thus, K_{min} tours (v_0, v_i, v_0) are obtained.

In the second step, a random unrouted node $v_j, j \neq 0$, is inserted in some route $tour_k$ at its best insertion position. The next two conditions must be hold – incorporation of v_j in this tour gives the least increase in a tour length among all other tours and demand of a potential node v_j does not exceed vehicle capacity. So:

- $\underset{\substack{v_a \in tour_k, \\ v_{a+1} \in tour_k, \\ v_j \notin tour_k}}{\operatorname{argmin}}\ c(v_a, v_j) + c(v_j, v_{a+1}) - c(v_a, v_{a+1})$;
- $D_{tour_k} + d_j \leq C$, where D_{tour_k} is a total demand of current $tour_k$.

If all conditions hold then this unrouted vertex $v_j, j \neq 0$ is inserted in a tour $tour_k$ between v_a and v_{a+1}.

The second step is repeated until no more unrouted vertex $v_j, j \neq 0$, can be feasibly inserted in some route $tour_k$. In this case a new tour $tour_k, k \leq K$, is initialized as (v_0, v_j, v_0) and adds to set of all tours $Tours$, and the procedure continues.

3.3 Nearest Neighbor heuristic (NN)

Nearest Neighbor heuristic constructs routes subsequently, one after another, in a greedy way.

In the first step, an unrouted node $v_i, i \neq 0$, which is closest to the depot v_0, is chosen. A new open tour $tour_k, k \leq K$, is initialized with v_i and v_0. Thus, a tour (v_0, v_i) is obtained.

In the second step, another unrouted vertex $v_j, j \neq 0$, is chosen, which is the nearest to the last added vertex and a demand of a potential node v_j does not exceed vehicle capacity. So, the next two formulae must be hold:

- $\underset{v_i \in tour_k,\ v_j \notin tour_k}{\text{argmin}} \quad c(v_i, v_j)$;
- $D_{tou\quad k} + d_j \leq C$, where D_{tour_k} is a total demand of current $tour_k$.

If all conditions hold then this unrouted vertex v_j is added in the end of $tour_k$ after v_i, and since that time it turns to be the last added vertex.

The second step is repeated until no more unrouted vertex $v_j, j \neq 0$, can be feasibly inserted. In this case a new tour $tour_k, k \leq K$, is initialized, and the procedure starts from the first step.

3.4 Clarke and Wright Savings heuristic (CWS)

In the first step, all vertices $v_i \in V, i \neq 0$, must form $|V - 1|$ routes. Thus, $|V - 1|$ tours (v_0, v_i, v_0) are obtained.

In the second step, $\forall v_i \in V, \forall v_j \in V, i \neq 0, j \neq 0, i \neq j$, saving $s(v_i, v_j)$ is calculated as $s(v_i, v_j) = c(v_i, v_0) + c(v_0, v_j) - c(v_i, v_j)$. All savings are put in a list of $\bar{S}$, $\bar{S}$ must be sorted in a non-increasing order.

In the third step, the first unused saving in a list is taken. Then, existence of two routes $tour_x$ and $tour_y$, $x \neq y$, having the next conditions, is checked:

- there is an edge (v_i, v_0) in route x and edge (v_0, v_j) in tour $tour_y$;
- $D_{tour_x} + D_{tou\quad y} \leq C$.

If there are such routes then $tour_x$ and $tour_y$ are combined by removing edges (v_i, v_0), (v_0, v_j) and introducing edge (v_i, v_j). After that, despite of ability or absence these routes, the current saving is skipped and the next one in the list is checked.

The last step works until K tours are left.

3.5 Variant of Clarke and Wright Savings heuristic (CWS_2)

Classical variant of Clarke and Wright Savings algorithm forms good tours in the first part of its work mostly. However, it was noticed that it tends to produce less competitive tours towards the end because of periphery nodes addition. Thus, Yellow [11] and Gaskell [12] suggested improved form of savings calculation. It is $s(v_i, v_j) = c(v_i, v_0) + c(v_0, v_j) - \lambda c(v_i, v_j)$. Here λ is a parameter which responds for measuring the distance between the vertices to be joint. In one report [13] it was mentioned that the best value of λ is 0.4.

3.6 Subgroup of Cluster-First-Route-Second heuristics

Subgroup of Cluster-First-Route-Second heuristics belongs to two-phase methods, which are based on the decomposition of the CVRP solution process into two separate stages – clustering and routing. In the clustering stage, a partition of the customers into routes is made, and in the routing stage, the sequence of the customers on each subset is obtained.

In Cluster-First-Route-Second methods, nodes are first partitioned into different subsets called clusters and then routes are determined by sequencing the customers within each subset.

3.6.1 Sweep

This Cluster-First-Route-Second method can be applied only for planar instances [9].

Clustering stage

Let us define $v_i \in V$ as $v_i = (x_i; y_i)$, where x_i and y_i are the Cartesian coordinates of point v_i.

In the first step, new normalized vertices $v_i' = (x_i'; y_i') = (x_i - x_0; y_i - y_0)$ are introduced, where the depot v_0' has new Cartesian coordinates (0; 0), $\forall i \in |V|$.

Let $\overline{v_i'} = (\theta_i, r_i)$ be a vertex with polar coordinate of v_i', where $r_i = x_i'^2 + y_i'^2$ and θ_i is calculated using formula 3:

$$\theta_i = \begin{cases} arctg\left(\frac{y_i}{x_i}\right), x_i > 0, y_i \geq 0 \\ arctg\left(\frac{y_i}{x_i}\right) + 2\pi, x_i > 0, y_i < 0 \\ arctg\left(\frac{y_i}{x_i}\right) + \pi, x_i < 0 \\ \frac{\pi}{2}, x_i = 0, y_i > 0 \\ \frac{3\pi}{2}, x_i = 0, y_i < 0 \end{cases} \quad (3)$$

In the second step, a list $\bar{V}$ of all $\overline{v_i'} = (\theta_i, r_i), \forall i \in |V|, i \neq 0$, is calculated and is sorted in increasing order by parameter θ_i.

In the next step, a new cluster is initialized with $\{v_0\}$ and maximum number of first L vertices from $\bar{V}$, such that $\sum_{i=0}^{L-1} d_i \leq C$. Parameter L is not a constant, it can be other for different clusters depending on weights of demands and total capacity. Then these used vertices are removed from $\bar{V}$, and the procedure is repeated until $\bar{V} = \emptyset$.

Routing stage

At this stage for each cluster TSP Cheapest Insertion heuristic is applied which forms a cycle.

3.6.2 Fisher and Jaikumar algorithm

In contrast to Sweep algorithm, this Cluster-First-Route-Second method can be applied not only for planar instances. Instead of using a geometric method to form the clusters, it solves a Generalized Assignment Problem (GAP).

Clustering stage

In the first step $\forall k = \overline{1..K}$ a vertex $v_{seed(k)} \in V \backslash \{v_o\}$ is chosen. These K vertices form K clusters.

In the second step the cost $cost_{v_i}^k$ of allocating each node $v_i \in V, i \neq 0$, to each cluster k is calculated as $cost_{v_i}^k = c(v_o, v_i) + c(v_i, v_{seed(k)}) - c(v_{seed(k)}, v_0)$.

In the third step the algorithm solves GAP with $cost_{v_i}^k$, d_i and C, which determines a minimum cost assignment of items to a given set of bins of capacity C. The GAP can be solved using either exact or heuristic techniques.

Routing stage

The final routes are determined by solving a TSP on each defined cluster.

According to this work [15], this algorithm gives way to the algorithms described above and provides solutions with more solution quality. That is why it will not be considered in later comparison study as it was already done.

3.7 Subgroup of Route-First-Cluster-Second heuristics

Subgroup of Route-First-Cluster-Second heuristics also belongs to two-phase methods. However, in contrast to Cluster-First-Route-Second methods, these constructive heuristics at first solve TSP for all nodes and only then break built cycle to K routes. Unfortunately, many studies showed than these heuristics are applicable only if there is no constraint on the number of vehicles. In addition, they are not competitive with other constructive heuristics in general [9].

4. Experiments and results

All algorithms are implemented as sequential algorithms in C++. The computational testing of the solution methods for CVRP has been carried out by considering eight sets of test instances from the next well-known database [16]. Total number of instances in sets A, B, E, F, G, M, P, X is 211. All instances inside one set have its own characteristics and a way of generation: cluster-based / uniform / geometric distribution of clients, real-world / imitative cases etc. The integer Euclidean metric is used for all instances. The naming scheme and data format for each instance is described here [17]. Shortly, the first letter in names shows the name of used set, the figure after letter 'n' shows the number of nodes and the figure which stands after letter 'k' presents the number of vehicles.

Experiment starts with the choice of a constructive heuristic H from the set {SI, FI, NN, CWS, CWS_2, Sweep}. After that one dataset D is selected from the list of all benchmark datasets. Then an instance file F from the chosen dataset D is taken as input for the algorithm H and the heuristic is executed (only 1 time because all these algorithms do not use random generations, so all obtained solutions are the same). After that we report solution quality $\varepsilon(\mathrm{H},\mathrm{F})$ found for the algorithm H on the test F. Solution quality ε (or percent above best-known, or gap) is calculated using formula 4 [18]:

$$\frac{F(S^0) - F_{opt}(S)}{F_{opt}(S)} \cdot 100\%, \tag{4}$$

where $F(S^0)$ is a length of obtained solution and $F_{opt}(S)$ is a length of optimal solution or best-known one. And finally, among all $\varepsilon(\mathrm{H},\mathrm{F})$ from one dataset sample mean $\bar{X}_\varepsilon(\mathrm{H},\mathrm{D}) = \frac{1}{|D|}\sum_{F=1}^{|D|}\varepsilon(\mathrm{H},\mathrm{F})$ is calculated which shows average gap for the algorithm H on the dataset D, where $|D|$ is a number of input files in dataset D.

The plan of experiments on constructive heuristics described in Fig. 1.

```
Input: constructive heuristics, datasets
      1:     foreach constructive heuristic H
      2:     foreach dataset D from datasets
      3:     foreach instance file F from D
      4:     solution = run H on F
      5:     calculate $\varepsilon(H,F)$
      6:     calculate $\bar{X}_\varepsilon(H,D)$ // average gap on dataset
```

Fig.1. Plan of experiment on constructive heuristics

It should be mentioned that each algorithm is subsequently launched on all 211 instances from 8 datasets, so no input file is missed.

A criterion of running time was not considered because all instances were solved in a time which does not exceed 1 second. It is thought to be insignificant in comparison with time-consuming metaheuristic work.

Fig. 2, 3 and 4 represent the results of experiments conducted over algorithms using sets B, P and G of widely different types. The horizontal axis represents the name of instance data. The vertical axis shows the solution quality.

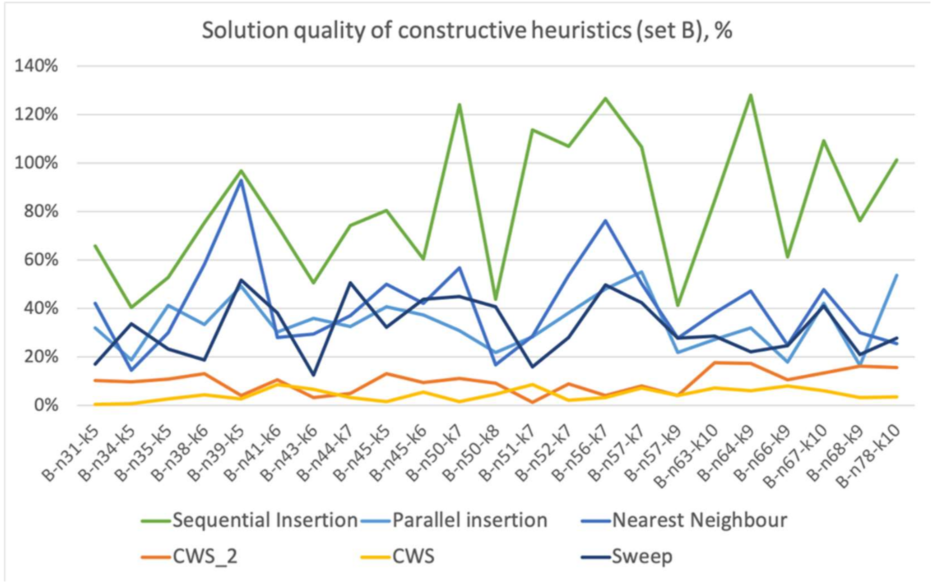

Fig. 2. Solution quality of constructive heuristics, set B

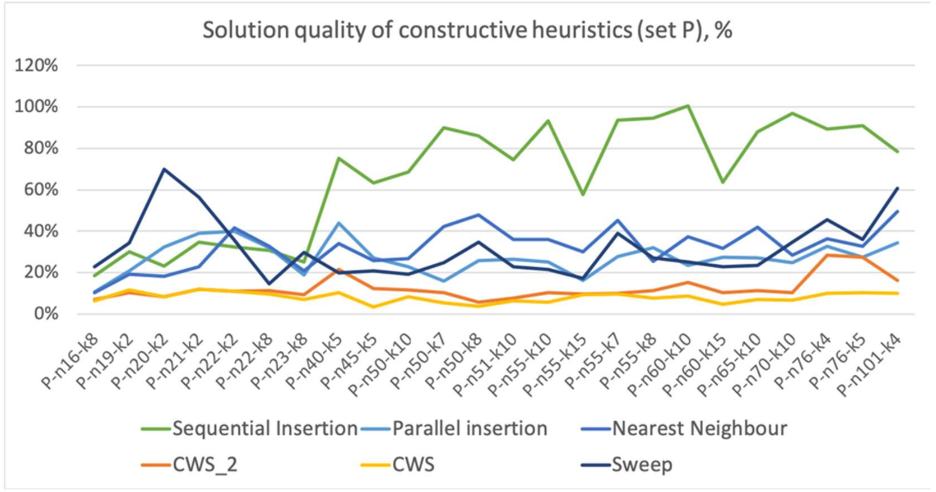

Fig. 3. Solution quality of constructive heuristics, set P

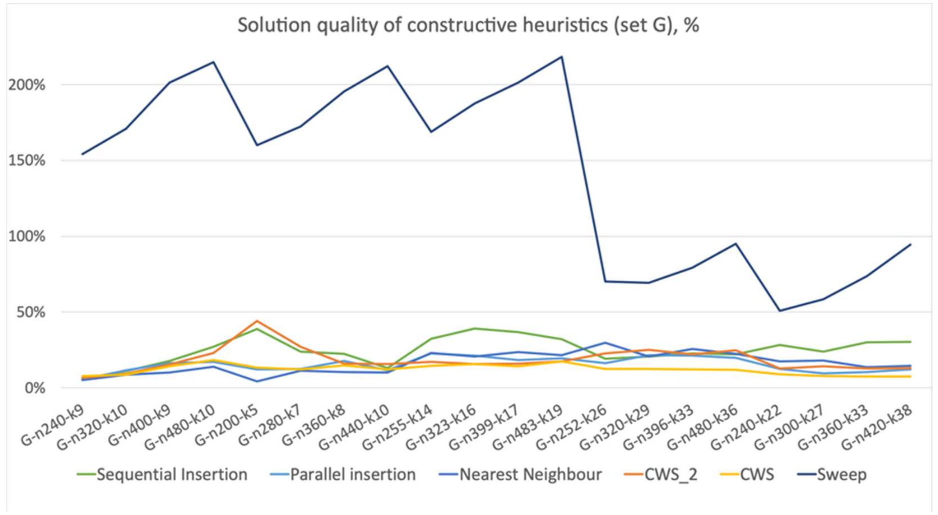

Fig. 4. Solution quality of constructive heuristics, set G.

Average gaps $\bar{X}_\varepsilon(\mathrm{H},\mathrm{D})$ of each algorithm on different data sets are presented in Table 1 and fig. 5. These general figures can show an approximate overall effectiveness of algorithms. On the basis of Table 1, all fig. 2, 3, 4, 5 and other results which cannot be shown here because of their large volume, it can be easily seen that CWS algorithm (its column is made bold in the table) is a leader for all input files, except some instances from dataset G. Its average gap varies from 3,4% till 11,0%. The closest competitor is its variant CWS_2, which has average solution quality in a range [9,8%; 20,6%]. CWS_2 algorithm is able to construct the best solutions only for some instances in set B. In all other cases this algorithm nearly always takes second place and goes behind classical CWS.

Table 1. Average gaps of all heuristics for every set, %.

Average gap $\bar{X}_\varepsilon(\mathrm{H},\mathrm{D})$ in the dataset		Constructive heuristic					
		SI	PI	NN	CWS	CWS_2	Sweep
Set (its size)	A (26)	68,7%	33,2%	39,7%	**5,0%**	12,7%	40,2%
	B (23)	82,3%	34,0%	41,2%	**4,3%**	9,8%	31,7%
	E (11)	70,4%	30,0%	41,5%	**6,4%**	17,5%	36,4%
	F (3)	42,5%	48,5%	74,6%	**4,4%**	20,6%	71,9%
	G (20)	24,8%	15,6%	16,3%	**11,0%**	18,4%	142,4%
	M (4)	83,0%	35,5%	44,6%	**3,4%**	12,0%	89,2%
	P (24)	66,0%	25,6%	32,2%	**6,9%**	11,3%	31,4%
	X (100)	99,7%	23,3%	27,4%	**5,9%**	11,9%	82,9%

Table 2. Percentage of unsolved instances for every set, %.

Percentage of unsolved instances in the set		Constructive heuristic					
		SI	PI	NN	CWS	CWS_2	Sweep
Set (its size)	A (26)	0%	0%	0%	0%	0%	**69,0%**
	B (23)	0%	0%	0%	0%	0%	**50,0%**
	E (11)	0%	0%	0%	0%	0%	**43,5%**
	F (3)	0%	0%	0%	0%	0%	**63,6%**
	G (20)	0%	0%	0%	0%	0%	**66,7%**
	M (4)	0%	0%	0%	0%	0%	**90,0%**
	P (24)	0%	0%	0%	0%	0%	**50,0%**
	X (100)	0%	0%	0%	0%	0%	**58,3%**

There is only one algorithm that have a problem with finding an answer to the given problems – it is Sweep. This heuristic is not able to construct a set of routes without exceeding the number of vehicles for some input files. All the others coped with the task – they are NN, SI, PI, CWS and CWS_2. Table 2 shows the percentage and the number of unsolved instances for all sets. In average, Sweep algorithm cannot solve the instance without over limit in more than 50% cases. It can be explained by the fact that the next vertex to be added is chosen by criteria of distance (polar angle, for real) but not the capacity.

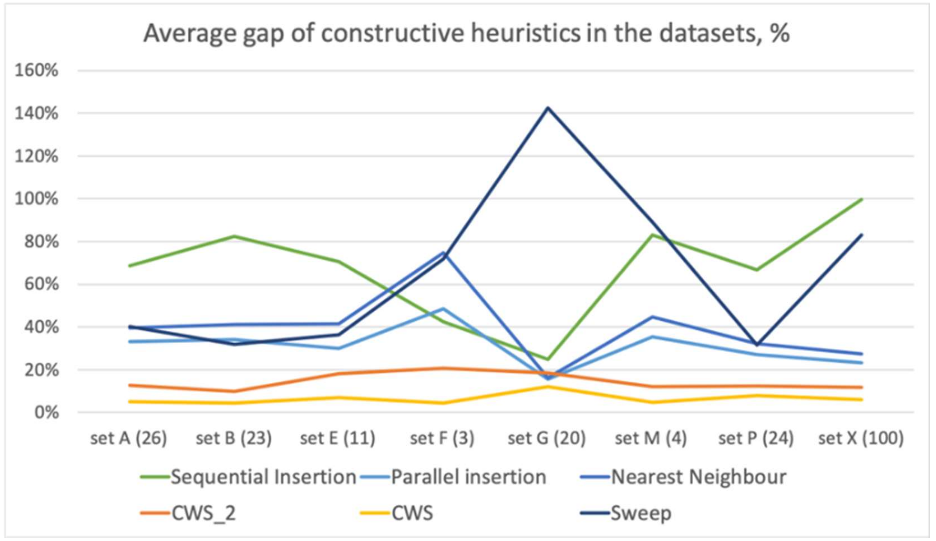

Fig. 5. Average gap of constructive heuristics in the datasets

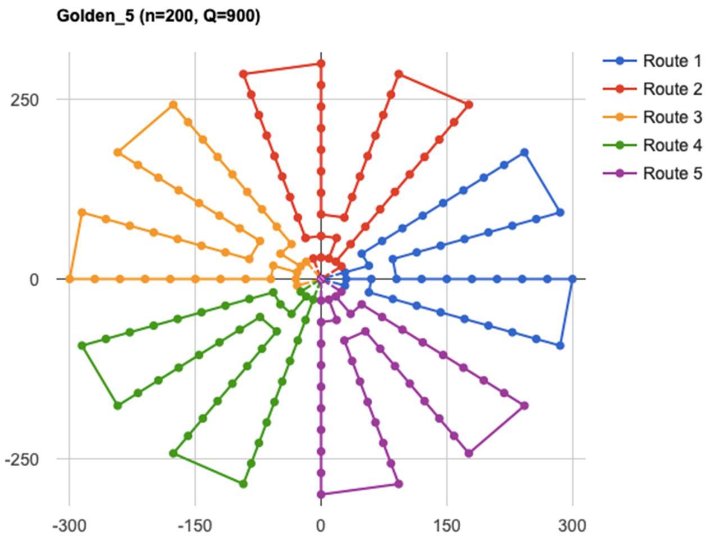

Fig. 6. Solution for instance G-n200-k5

It was mentioned earlier that CWS is not a leader for some instances from dataset G. There are 8 instances when NN finds the best solutions but not CWS (fig. 4). This interesting change of the leader is connected with the type of customers' distribution – these instances have a form of rays going from the center. If we look at fig. 6, where a solution for the instance is presented, we can see that the idea of nearest neighbor works here the best way.

5. Conclusions

Overall, the next recommendation should be given to the problem which has described variant of mathematical model of CVRP. In general, for all types of clients' distribution the best algorithm to be applied is Clarke and Wright Savings, however, in case of having input data in form of concentric rays (like in fig. 6) it is better to use Nearest Neighbor algorithm. Also, a few instances were solved best of all by Clarke and Wright Savings 2 algorithm, so it is important to have this algorithm in mind, however the difference between it and CWS is not very significant (no more than 1%). One more conclusion is that it is unreasonable to use Sweep heuristic as it is not able to construct a set of routes without exceeding the number of vehicles for more than 50% of input files. Finally, for our research it means that for all instances, except those 8 from set G, CWS heuristic will be used as initial algorithm for metaheuristic, otherwise – we will apply NN.

References

[1] P. Toth and D. Vigo, "Branch-and-Bound algorithms for the capacitated VRP," in The Vehicle Routing Problem, Philadelphia, SIAM, 2002, pp. 29-51.
[2] K. Braekers, K. Ramaekers, and I. Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. Computers & Industrial Engineering, vol. 99, 2016, pp. 300-313.
[3] B. Golden, S. Raghavan and E. Wasil. The vehicle routing problem: latest advances and new challenges. New York: Springer, 2008.
[4] P. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. Computers & Operations Research, vol. 34, no. 8, 2007, pp. 2403-2435.
[5] Y. Nagata and O. Braysy. Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. Networks, vol. 54, no. 4, 2009, pp. 205-215.
[6] T. Vidal, T. Crainic, M. Gendreau, N. Lahrichi, and W. Rei. A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems. Operations Research, vol. 60, no. 3, 2012, pp. 611-624.
[7] E. Beresneva and S. Avdoshin. Analysis of mathematical formulations of Capacitated Vehicle Routing Problem and methods for their solution. Trudy ISP RAN/Proc. ISP RAS, vol. 30, no. 3, 2018, pp. 233-250. DOI: 10.15514/ISPRAS-2018-30(3)-17.
[8] M. Reed and B. Simon. Methods of modern mathematical physics. London: Academic Press, 1972.
[9] G. Laporte and F. Demet. Classical heuristics for the Capacitated VRP. In The Vehicle Routing Problem, SIAM, 2002, pp. 109-128.
[10] G. Laporte, Y. Nobert, and M. Desrochers. Optimal routing under capacity and distance restrictions. Operations Research, vol. 33, no. 5, 1985, pp. 1050–1073.
[11] P. Yellow. A computational modification to the savings method of vehicle scheduling, Operational Research Quarterly, no. 21, 1970, pp. 281-283.
[12] T. Gaskell. Bases for vehicle fleet scheduling. Operational Research Quarterly, no. 18, 1967, pp. 281-295.
[13] B. Golden, T. Magnanti, and H. Nguyen. Implementing vehicle routing algorithms. Networks, no. 7, 1977, pp. 113-148.
[14] M. L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. Networks, vol. 11, no. 3, 1981, pp. 109-124.
[15] T. Sultana, M. Akhand and M. Rahman. A variant Fisher and Jaikuamr algorithm to solve capacitated vehicle routing problem. In Proc. of the 8th International Conference on Information Technology (ICIT), 2017, pp. 710-716.
[16] I. Xavier. CVRPLIB. [Online]. Available: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/. [Accessed 09 07 2019].
[17] Heidelberg University. TSPLIB. [Online]. Available: https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/. [Accessed 09 07 2019].

[18] P. Toth and D. Vigo. An overview of vehicle routing problems. In The Vehicle Routing Problem, SIAM, 2002.

Информация об авторах / Information about authors

Екатерина Николаевна БЕРЕСНЕВА – с 2017 года преподаватель департамента программной инженерии НИУ ВШЭ, с 2019 года – аспирант НИУ ВШЭ. Профессиональные интересы – дискретная математика, задача маршрутизации транспорта, задача коммивояжера.

Ekaterina BERESNEVA – lecturer at School of Software Engineering, Faculty of Computer Science, National Research University Higher School of Economics since 2017. Her research interests include discrete mathematics, the vehicle routing problem and the travelling salesman problem.

Сергей Михайлович АВДОШИН – профессор, руководитель департамента программной инженерии факультета компьютерных наук НИУ ВШЭ с 2005 года. Сфера научных интересов: разработка и анализ компьютерных алгоритмов, имитация и моделирование, параллельные и распределенные процессы, теневой интернет, технология блокчейн.

Sergey AVDOSHIN – Professor, Head of School of Software Engineering in National Research University Higher School of Economics since 2005. Research interests are design and analysis of computer algorithms, simulation and modeling, parallel and distributed processing, deep Web, blockchain technology.