ISSN 1990-4789, Journal of Applied and Industrial Mathematics, 2015, Vol. 9, No. 4, pp. 503–518. © Pleiades Publishing, Ltd., 2015.
Original Russian Text © Yu.A. Kochetov, A.V. Khmelev, 2015, published in Diskretnyi Analiz i Issledovanie Operatsii, 2015, Vol. 22, No. 5, pp. 5–26.

A Hybrid Algorithm of Local Search for the Heterogeneous Fixed Fleet Vehicle Routing Problem

Yu. A. Kochetov[1,2*] and A. V. Khmelev[2]**

[1] *Sobolev Institute of Mathematics, pr. Akad. Koptyuga 4, Novosibirsk, 630090 Russia*
[2] *Novosibirsk State University, ul. Pirogova 2, Novosibirsk, 630090 Russia*

Received March 13, 2015; in final form, June 15, 2015

Abstract—Under consideration is the optimization problem for the routes of heterogeneous vehicles serving a given set of customers. It is assumed that the customers are represented by points in the plane, whereas the number of each type of vehicles is limited. To solve the problem, we developed a hybrid local search algorithm with coding the solutions as a sequence of customers. To decode the sequence, the corresponding NP-hard problem is solved by the method of Lagrangian relaxation. We propose new procedures for intensification and diversification of the search, as well as a new neighborhood of exponential size. The results of computational experiments are presented for the available test instances with the number of customers up to 255. For 15 instances we obtain new record values of the objective function.

DOI: 10.1134/S1990478915040079

Keywords: *local search, exponential neighborhood, Lagrangian relaxation, subgradient optimization*

INTRODUCTION

The vehicle routing problem is a rapidly developing field of operations research. Owing to numerous applications, a profusion of new mathematical models appears, the original numerical methods are developed, and some specialized international conferences are convened. In the paper, we study the problem of finding optimal routes for heterogeneous vehicles of limited load-carrying capacity. It is assumed that each type of vehicles has a fixed cost and specific travel cost (the cost of gasoline per kilometer). The number of vehicles of each type is limited. All vehicles are in a garage (depot) and must return there after servicing customers. Each customer has a specific demand for the delivery of goods and is served by exactly one vehicle. The depot and customers are represented by points on the plane. It is required to find the routes of vehicles delivering for goods to all customers with a minimum total cost.

Along with this problem (Heterogeneous Fixed Fleet Vehicle Routing Problem, HFFVRP), under consideration are the version with the unlimited vehicle fleet, the mixed version, and the version with identical vehicles. These problems are all NP-hard and actively studied both theoretically and empirically in order to obtain efficient algorithms. A surwey of recent results can be found, for example, in [5].

In the development of algorithms for solving the routing problems, the following subproblem often arises: Some sequence of customers is given; we have to split it into feasible routes and assign the vehicles to them so that the total cost of serving the customers be minimal. If we can solve this subproblem, then the possibility appears to encode solutions in the form of a sequence of customers and use the well-acclaimed methods for these problems, in particular, for the traveling salesman problem. With an unlimited vehicle fleet, this subproblem can be solved in polynomial time. With a limited fleet, it is NP-hard [20] but can be solved exactly by the method of dynamic programming.

In this paper, we use an approximation method of Lagrangian relaxation which yields some upper and lower bounds for the optimum. This method is used in the scheme of local search with variable

* E-mail: jkochet@math.nsc.ru
** E-mail: avhmel@gmail.com

neighborhoods. Along with the well-known neighborhoods, we introduce a new neighborhood of exponential size, which allows us to make a deep restructuring of solutions with improvement of the objective function. Some new procedures for intensification and diversification of local search are applied to improve the efficiency of the method. The results of computational experiments testify the high competitiveness of the new hybrid method. For 15 test instances from the widely-used electronic collections we improved the record values of the objective function.

In Section 1, the exact formulation of the problem is given and a brief overview of the previous research is presented. In Section 2, we consider the problem of splitting a sequence of customers into the vehicle routes and the method of Lagrangian relaxation to solve it. The developed local search algorithm is described in Section 3, and the results of computational experiments are presented in Section 4. The article is completed with a conclusion and a description of the areas of future research.

1. STATEMENT OF THE PROBLEM

Consider a complete undirected weighted graph $G = (V, A)$ with the vertex set $V = \{0, 1, \ldots, n\}$ and the arcs $A = \{(i, j) \mid i, j \in V, i < j\}$. Vertex 0 corresponds to a depot, where the vehicles are placed. The other vertices symbolize the customers: $V' = V \setminus \{0\}$. All vertices are represented by points on the plane. The distance d_{ij} is given between two of vertices (i, j). Each customer $i \in V'$ has an order for the delivery of q_i units of cargo. The vehicle fleet consists of heterogeneous vehicles. Let the set of types of vehicles be denoted by K. For each $k \in K$, the number m_k of available vehicles and their limit capacity Q_k are known. The use of a vehicle entails a one-time fixed costs f_k. The ride from i to j costs $c_{ij}^k = d_{ij}c_k$, where c_k is the cost per unit distance for a vehicle of type k.

Let r be a route in G. This route is said to be *feasible for a vehicle of type* k if it forms a simple circuit passing through vertex 0 and the total demand of customers does not exceed the capacity of the vehicle. The cost of the route r is formed by the fixed cost f_k of the vehicle associated with r and the integrated cost of the arcs. The problem is to find a set of feasible routes and to assign vehicles to them so that each customer is visited exactly once, the number of vehicles involved does not exceed the size of the vehicle fleet, whereas the total cost of the routes is minimal.

Let us give the exact mathematical formulation of the problem. We introduce the variables:

$x_{ij}^k = 1$ if a vehicle of type k travels from customer i to customer j; and $x_{ij}^k = 0$ otherwise;

$y_{ij} \geq 0$ is the quantity of goods in the vehicle when it goes from customer i to customer j.

Using these variables, we can state the HFFVRP problem in terms of mixed integer linear programming:

$$\min \Big(\sum_{k \in K} f_k \sum_{j \in V'} x_{0j}^k + \sum_{k \in K} \sum_{i,j \in V} c_{ij}^k x_{ij}^k \Big), \tag{1}$$

$$\text{s. t.} \quad \sum_{k \in K} \sum_{i \in V} x_{ij}^k = 1, \qquad j \in V', \tag{2}$$

$$\sum_{i \in V} x_{ij}^k = \sum_{i \in V} x_{ji}^k, \qquad j \in V, \quad k \in K, \tag{3}$$

$$\sum_{j \in V'} x_{0j}^k \leq m_k, \qquad k \in K, \tag{4}$$

$$\sum_{i \in V} y_{ij} - \sum_{i \in V} y_{ji} = q_j, \qquad j \in V', \tag{5}$$

$$y_{0j} \leq \sum_{k \in K} Q_k x_{0j}^k, \qquad j \in V', \tag{6}$$

$$y_{ij} \leq \sum_{k \in K} (Q_k - q_i) x_{ij}^k, \qquad i \in V', \quad j \in V, \quad i \neq j, \tag{7}$$

$$x_{ij}^k \in \{0, 1\}, \qquad y_{ij} \geq 0, \qquad i, j \in V, \quad k \in K. \tag{8}$$

Table 1. Feasible and infeasible sequences

$m_1 = 1,\ Q_1 = 10,$						$m_2 = 1,\ Q_2 = 14$	
r	:	a	b	c	d	e	$\rightarrow$ feasible
q	:	5	4	4	2	7	
r'	:	a	d	c	e	b	$\rightarrow$ infeasible
q	:	5	2	4	7	4	

The objective function of the problem determines the total cost of engaging the vehicles and delivering all goods to the customers. Constrains (2) and (3) ensure that each customer is visited exactly once and a vehicle leaves the customer after visiting. Constrains (4) set an upper bound on the number of the engaged vehicles of each type. Constrain (5) is a restriction on the commodity flow: the difference in the quantity of goods in a vehicle before and after visiting some customer must be equal to the customer's demand. Moreover, this constrain excludes the cycles that are not passing through vertex 0. Constrain (6) guarantees that the vehicle capacity is never exceeded, whereas (7) establishes a relationship between the variables y_{ij} and x_{ij}^k: if none of vehicles uses the edge (i, j) then the cargo flow on it equals zero. The domain of definition of the variables is defined by (8).

The HFFVRP problem was firstly considered in [22], where the method of column generation combined with a local search procedure and a procedure of adaptive memory control was proposed to solve the problem. An exact method based on reducing the problem to the set partitioning problem, was developed in [6]. This method can find the optimal solutions for instances with up to 100 customers and, as of today, it is the best exact approach. The hybrid iterative local search method presented in [21] is based on the same reducing. In this method, the set partitioning problem is solved many times by an optimization software package for integer linear programming problems. The threshold algorithms (simulated annealing, record to record travel algorithms) were investigated in [14, 23, 24]. Some variants of tabu search with various diversification mechanisms were suggested in [7, 9, 19, 25]. The idea of variable neighborhoods was used in [15, 16]. New neighborhoods, in particular, the neighborhoods of exponential size, as well as their random ordering allowed us to improve the record values of the objective function for many test instances.

The problem of splitting the sequence of customers into routes was used in several algorithms. In [20], an exact method of dynamic programming was used for solving it. In [11–13], this method is used for obtaining fast approximate solution. This same idea was successfully applied to the split delivery vehicle routing problems [7] and the vehicle routing problems with time windows [10]. In [26], the general scheme is proposed for solving the routing problems which includes a generalized partition algorithm. This scheme was applied to 29 different types of the problem and proved the effectiveness.

In the case of limited heterogeneous vehicle fleet, the problem becomes much more complicated. There are such sequences for which no feasible solution exists (see Table 1). As was already mentioned in the Introduction, the problem becomes NP-hard. Therefore, in this paper we use a different approach that is based on the Lagrangian relaxation implemented in the subgradient optimization method.

2. THE METHOD OF THE LAGRANGIAN RELAXATION FOR SPLITTING A SEQUENCE

Let us represent the problem of splitting the sequence $\pi = (\pi_1, \ldots, \pi_n)$, $\pi_i \in V'$, into routes in terms of finding the shortest path in a directed weighted multigraph. The vertex π_1 is the only source, whereas the vertex $n + 1$, the only sink. Given (i, j), $1 \le i < j \le n + 1$, we define the set of arcs $(i, j)^k$, $k \in K$, corresponding to the vehicle of type k coming from the depot to customer π_i, then to customer π_{i+1}, and so on to customer π_{j-1}, and then returning back to the depot. The cost L_{ij}^k of such a path is defined as follows:

$$L_{ij}^k = \begin{cases} f_k + r_k\Big(d_{0\pi_i} + \sum\limits_{t=i}^{j-2} d_{\pi_t\pi_{t+1}} + d_{\pi_{j-1}0}\Big), & \text{if } \sum\limits_{t=i}^{j-1} q_{\pi_t} \le Q_k, \\ \infty, & \text{otherwise.} \end{cases}$$

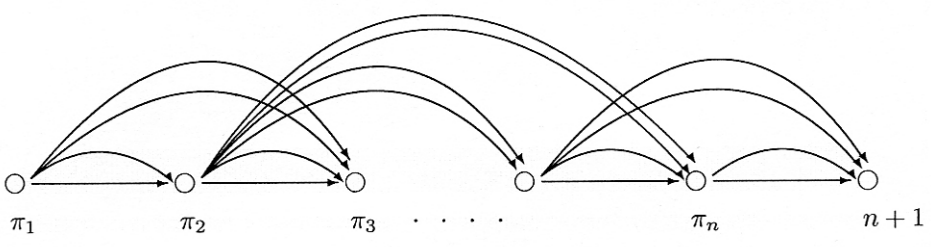

Fig. 1. A multigraph for the sequence π.

An optimal sequence splitting can be obtained by finding a path of the minimum cost from source to sink, while limiting the number of the used arcs of each type (Fig. 1).

Let us write this problem in terms of integer linear programming. Let a Boolean variable z_{ij}^k take the value 1 if and only if the arc $(i,j)^k$ is included into the optimal solution. Then the required path can be obtained by solving the following problem:

$$\min \sum_{k\in K}\sum_{i=1}^{n}\sum_{j=i+1}^{n+1} L_{ij}^k z_{ij}^k, \tag{9}$$

$$\text{s. t.} \sum_{k\in K}\sum_{j=i+1}^{n+1} z_{ij}^k - \sum_{k\in K}\sum_{j=1}^{i-1} z_{ji}^k = \begin{cases} 1, & \text{if } \ i=1, \\ -1, & \text{if } \ i=n+1, \\ 0, & \text{if } \ 1<i<n+1, \end{cases} \tag{10}$$

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n+1} z_{ij}^k \le m_k, \qquad k\in K, \tag{11}$$

$$z_{ij}^k \in \{0,1\}, \qquad k\in K, \qquad 1\le i<j\le n+1. \tag{12}$$

The objective function (9) defines the total cost of the path from source to sink. Constraint (10) ensures that the number of incoming arcs coincides with the number of outgoing arcs at each vertex which is not the source or the sink. A single arc comes from the source, and a single arc comes to the sink. Constraint (11) sets an upper bound for the number of vehicles of each type in use.

Problem (9)–(12) is NP-hard; however, the removal of (11) makes it polynomially solvable. We introduce the nonnegative Lagrange multipliers λ_k for each of the inequalities (11) and add the term

$$\sum_{k\in K}\lambda_k\left(\sum_{i=1}^{n}\sum_{j=i+1}^{n+1} z_{ij}^k - m_k\right)$$

into the objective function (9). Collecting similar terms, we obtain the relaxed problem:

$$LR(\lambda_k) = \min \sum_{k\in K}\sum_{i=1}^{n}\sum_{j=i}^{n} \left(L_{ij}^k + \lambda_k\right) z_{ij}^k - \sum_{k\in K}\lambda_k m_k$$

subject to (10) and (12).

Note that this problem is the shortest path problem in the multigraph for the sequence π (see Fig. 1) with the arcs of weights

$$\overline{L}_{ij}^k = L_{ij}^k + \lambda_k.$$

It can be easily and exactly solved by dynamic programming and gives a lower bound of the objective function (9). The search for the best of these estimates over the Lagrange multipliers leads to the dual problem:

$$D = \max_{\lambda_k\ge 0} LR(\lambda_k).$$

To solve it, we use the methods of subgradient optimization. Take some initial vector λ^0 (for example, the zero) and, at the next iteration t, change the multipliers λ^t according to the rule

$$\lambda^{t+1} := \max\left\{0,\ \lambda^t + h^t\left(\sum_{i=1}^{n}\sum_{j=i+1}^{n+1} z_{ij}^k(\lambda^t) - m_k\right)\right\},$$

where $z_{ij}^k(\lambda^t)$ is an optimal solution of problem $LR(\lambda^t)$ and h^t is the step-size in the subgradient direction

$$\Delta_k^t = \sum_{i=1}^{n}\sum_{j=i+1}^{n+1} z_{ij}^k(\lambda^t) - m_k, \qquad k \in K,$$

of the objective function of the dual problem at $\lambda = \lambda^t$. If $h^t \to 0$ and $\sum_{t=0}^{\infty} h^t \to \infty$ then $LR(\lambda^t) \to D$ [17]. The application of such divergent series leads to large computational cost and slow convergence to the optimum. In practice, other rules are applied that ensure rapid convergence, but do not guarantee optimality. Following [4], we move not along subgradient, but along the difference of two successive subgradients:

$$\lambda^{t+1} := \max\left\{0,\ \lambda^t + \alpha h^t\Delta^t - (1-\alpha)h^{t-1}\Delta^{t-1}\right\},$$

where α is chosen in the interval from 0 to 1. The step-size h^t is calculated from the subgradient Δ^t, the value of the upper bound of the optimum UB, and the parameter Θ^t:

$$h^t := \Theta^t(UB - LR(\lambda^t))/\|\Delta^t\|^2.$$

The factor Θ^t is firstly taken as a positive constant not greater than 2 and then divided by a quantity exceeding 1 (after a given number of iterations which depends on the problem dimension).

If at some iteration the subgradient turns out to be nonpositive, $\Delta^t \le 0$, then the solution $z_{ij}^k(\lambda^t)$ is feasible, and the value of the objective function (9) on this solution provides an upper bound for the optimum UB. Moreover, if

$$\sum_{k\in K} \lambda_k^t\Delta_k^t = 0$$

then the values of the upper and lower bounds coincide, and the solution $z_{ij}^k(\lambda^t)$ is optimal. The duality gap in this case is absent, which is observed rather rarely. Nevertheless, this approach rapidly yields the approximate solutions with a small relative error, whereas, with nonrigid restrictions on the number of the vehicles involved, it often finds the exact solution of the problem. At that, the equality

$$\sum_{k\in K} \lambda_k^t\Delta_k^t = 0$$

may fail to hold.

3. LOCAL SEARCH

To find the best customers sequence, we will use the metaheuristics that have proven efficiency in solving the problems of this type [13, 20, 26]. One of these metaheuristics is the local search with variable neighborhoods [2, 8]. The successful implementation of this scheme requires a set of various neighborhoods. They will contain only feasible solutions.

Let π be a permutation, let $R(\pi) = (r_1, \ldots, r_{|R|})$ be a set of routes, and let some type of vehicle be assigned to each of them. Given $R(\pi)$, the solution of the problem (1)–(8) is reconstructed uniquely.

Consider five interroute neighborhoods in which at most two customers are moved between two routes [15]:

Shift$(1, 0)$. One customer is moved to the best position in another route.

Swap$(1,1)$. Two customers from different routes swap their routes and occupy the best positions there.

Shift$(2,0)$. Two adjacent customers are transferred from one route to another and inserted there into the best position, possibly changing the order.

Swap$(2,1)$. Two adjacent customers from one route are transferred to another and are placed into the best position after removal of one of the customers of this route. The removed customer is transferred to the original route and is placed there in the best position. As in the previous neighborhood, for the pair of the moved customers, the two options of their placement are considered: according to the original and the reverse order.

Swap$(2,2)$. Two adjacent customers from one route are replaced by two neighboring customers from another. The four variants of the order of these pairs are checked, and for each option the best position is selected for insertion into the route.

Analogous intraroute neighborhoods for the customers in the case of a single route will be denoted by $\text{Shift}_1(1,0)$, $\text{Swap}_1(1,1)$, $\text{Shift}_1(2,0)$, and $\text{Swap}_1(2,1)$.

The following neighborhood changes the structure of two routes by breaking them into four segments:

Cross. We select two routes. Each of them is divided into two nonempty ordered sets of customers, which are then concatenated to form two new routes. All variants of such concatenation are considered.

Finally, the last neighborhood concerns only one route and is widely applied to the traveling salesman problem:

2-opt. The route is represented as a cycle from which two nonadjacent arcs are removed and two other arcs are added so that a cycle is again formed.

While defining the neighborhoods, we assumed that the vehicle type does not change in restructuring the routes, which can lead to infeasible solutions or solutions with too high cost. Optimization of the vehicle fleet can improve this shortcoming, but this requires to solve problem (9)–(12). This results in a very time-consuming procedure. In order to reduce the complexity, the neighboring solutions are first considered without optimization of the vehicle fleet. If there are solutions with a smaller value of the objective function then the exploration of the neighborhood stops (the first improvement rule). Otherwise, we check the neighboring solutions again, but this time with the fleet reassignment.

The above-presented neighborhoods are used in the hybrid local search algorithm (HLS) whose main elements are: randomized variable neighborhood descent (RVND), the procedures of intensification (Intensify), diversification of the search (Perturb), and post-optimization (Postoptimize). This is an algorithm for obtaining local optima over the neighborhoods. However, we search these local optimum not at arbitrary locations in the feasible search domain, but next to the best obtained solution [2]. In the search diversification procedure, this solution is slightly modified in order to find a good start solution for a new local search. In the post-optimization procedure, an attempt is made to improve the obtained local optimum by using the neighborhoods of exponential size [1, 18]. A general scheme of the algorithm is shown below. The best local optimum found is denoted by π^*, whereas the smallest value of the objective function, by f^*. In the algorithm, the start solution π^0 is selected I times, and the RVND procedure is applied I_{LS} times.

Hybrid local search algorithm (HLS)

1. $f^* \leftarrow \infty$
2. Cycle over $i := 1, \ldots, I$
3. $\pi \leftarrow \pi^0$; $j \leftarrow 0$;
4. while $j \le I_{LS}$
5. $\pi' \leftarrow \text{RVND}(\pi)$
6. If $f(\pi) < f(\pi')$ and $j > I_{LS}/2$ then
7. $\pi' \leftarrow \text{Intensify}(\pi')$;
8. If $f^* > f(\pi')$ then $f^* \leftarrow f(\pi')$, $\pi^* \leftarrow \pi'$;
9. $\pi \leftarrow \text{Perturb}(\pi')$; $j = j + 1$;

10. $\pi^* \leftarrow$ Postoptimize(π^*);

11. Return π^* and f^*.

We construct the initial solution π^0 (line 3) using the following randomized procedure [15]: For each vehicle, we generate a route and insert a single customer to it. This customer is selected randomly with uniform distribution. Different routes get different customers. Next we select one of the two rules to insert the remaining customers into the routes and one of the two strategies, sequential or parallel. In the sequential strategy, one of the routes is randomly selected and is filled with customers, until it is allowed by the load-carrying capacity of the vehicle. In the parallel strategy, all routes are considered simultaneously, and the insertion of customers is performed into the route with minimal additional costs. To select the next customer j, the following two rules of the cost estimation are considered: either the lowest cost of traveling to the customer from the customers already included in the route (i.e., $\min\limits_i c^k_{ij}$) or

$$\min_i \left[\left(c^k_{ij} + c^k_{ji'} - c^k_{ii'}\right) - \gamma\left(c^k_{0j} + c^k_{j0}\right)\right].$$

In the last expression, we estimate the costs relative to the cost of the new route with the only customer j. The parameter γ for each customer is randomly selected from the interval $[0, 1.7]$.

The local search over the above interroute neighborhoods is performed by the RVND procedure. At each iteration of this procedure, a local optimum is obtained over the neighborhoods $\mathrm{Shift}_1(1,0)$, $\mathrm{Swap}_1(1,1)$, $\mathrm{Shift}_1(2,0)$, $\mathrm{Swap}_1(2,1)$, and 2-opt. The remaining neighborhoods are used to restructure these optima and obtain the starting point for the next iteration. The order of using these neighborhoods is random.

The general scheme of RVND is as follows:

Let $N = \{N_1, \ldots, N_l\}$ be a set of neighborhoods for restructuring the routes, and let π be some feasible solution of the problem. At each iteration, we randomly select one of the neighborhoods and find there the best solution π' among the neighbors of π. If π has turned out to be a local optimum (i.e., π' is not better than the original π) then we remove this neighborhood from N. Otherwise, we apply to π' the local improvement algorithm over the intra-route neighborhoods $\mathrm{Shift}_1(1,0)$, $\mathrm{Swap}_1(1,1)$, $\mathrm{Shift}_1(2,0)$, $\mathrm{Swap}_1(2,1)$, and 2-opt to improve the modified routes and find a new local optimum. Note that, at each iteration, either a new local optimum is found or the collection of neighborhoods is reduced.

In the process of intensification of the search over the set of routes R, we construct a customer sequence π for problem (9)–(12). The sequence is constructed as follows: The first route is selected randomly and added as a term into the sequence. Then the next route is selected, whose one of the extreme customers is closest to the last customer in the sequence. All customers from this route are added to the sequence. This process continues until all routes are used.

In the diversification procedure, we use the same method to construct a sequence of customers. Then L times we randomly select two elements of the sequence and reverse the order of elements between them (an analog of the 2-opt neighborhood moves). we solve problem (9)–(12) for the so-obtained sequence. If the solution to the problem is not feasible then we apply an alternative diversification procedure: we do L random neighborhood moves with $\mathrm{Swap}(1,1)$, $\mathrm{Shift}(1,0)$, or $\mathrm{Shift}(2,0)$ neighborhoods. A similar procedure was used in [15, 16].

The final local optimum is subjected to the post-optimization procedure (line 10). Here the neighborhoods of exponential size are used. Below we give description of two such neighborhoods based on the idea of ejection chains. The first neighborhood was previously used in [18], and the second is new.

As before, let $R(\pi) = (r_1, \ldots, r_{|R|})$ be a set of routes. Consider two elements π_i and $\pi_{i'}$ from different routes r and r'. By the *ejection move* we understand the removal of π_i from r, the removal of $\pi_{i'}$ from r', and its insertion of π_i at the best position in r' without $\pi_{i'}$. Let $w(\pi_i, \pi_{i'})$ define the cost of this restructuring of the routes r and r'; i.e.,

$$w(\pi_i, \pi_{i'}) = \left(c_{\pi_{i-1}\,\pi_{i+1}} - c_{\pi_{i-1}\,\pi_i} - c_{\pi_i\,\pi_{i+1}}\right) + \left(c_{\pi_{i'-1}\,\pi_{i'}+1} - c_{\pi_{i'-1}\,\pi_{i'}} - c_{\pi_{i'}\,\pi_{i'+1}}\right) + \left(c_{\pi_l\,\pi_i} + c_{\pi_i\,\pi_{l+1}} - c_{\pi_l\,\pi_{l+1}}\right)$$

if π_i is inserted in r' between π_l and π_{l+1} (Fig. 2).

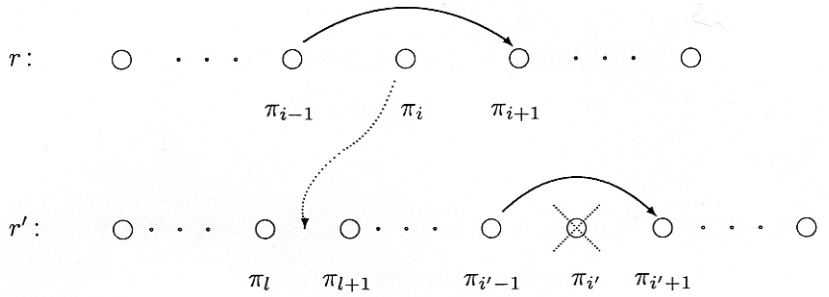

Fig. 2. Extraction for $\pi_i \in r$ and $\pi_{i'} \in r'$.

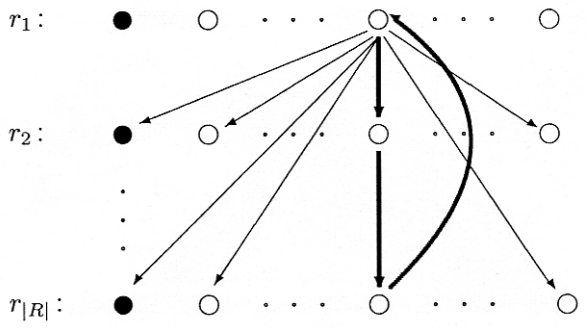

Fig. 3. Layered graph.

The *insertion move* is the operation of removing π_i from r and adding π_i at the best position in r' without forcing any ejection from it. For this operation, we define $w(\pi_i, r')$ as the cost increase of the route r' together with the saving that resulted by the removal of π_i from r, i.e.,

$$w(\pi_i, r') = \left(c_{\pi_{i-1}\,\pi_{i+1}} - c_{\pi_{i-1}\,\pi_i}\right) + \left(c_{\pi_{i'-1}\,\pi_{i'+1}} - c_{\pi_{i'-1}\,\pi_i} - c_{\pi_{i'}\,\pi_{i'+1}}\right) + \left(c_{\pi_l\,\pi_i} + c_{\pi_i\,\pi_{l+1}} - c_{\pi_l\,\pi_{l+1}}\right).$$

Note that $w(\pi_i, \pi_{i'})$ and $w(\pi_i, r')$ may be negative.

Based on these moves, we define an *ejection-insertion graph* whose vertices correspond to the routes and customers, and the arcs, to the transitions of ejection and insertion moves. This graph has a layered structure (Fig. 3). The layer contains a dummy vertex (sink) corresponding to the route itself and one vertex for each customer on the route.

Each interlayer arc has one of the two types in dependence on the move: the ejection arc $(\pi_i, \pi_{i'})$ with weight $w(\pi_i, \pi_{i'})$ and the insertion arc (π_i, r') with weight $w(\pi_i, r')$. The endvertices of the insertion arcs are dummy. In the case when the restrictions on the vehicle capacity are violated in these moves, the weight of the arc is assumed to be a sufficiently large number. If there is a cycle in this graph or a path from some vertex to the dummy vertex which passes through each layer at most once and has negative weight then the current solution can be rebuilt with some decrease in cost. Note that in the case of finding a path, the dummy vertex can be only at its end.

The problem of finding such a path or cycle is NP-hard. An approximate solution can be obtained by a modification of the Floyd–Warshall algorithm [18]. The modification consists in the following extra check: Assume that there are two paths: from vertex π_i to pi_k and from π_k to π_j. A path from π_i to π_j via π_k is feasible only if the paths from π_i to π_k and from π_k to π_j have no vertices from the same layer.

The second neighborhood used in the post-optimization process, involves splitting the routes into two parts and concatenating the parts of different routes. The difference between these neighborhoods is shown in Fig. 4.

Again, we construct a layered graph. Each layer corresponds to one route, but now it does not contain dummy vertices. Let the first layer correspond to the route $r_1 = (u_1, u_2, \ldots, u_{r_1})$. Since the route can be

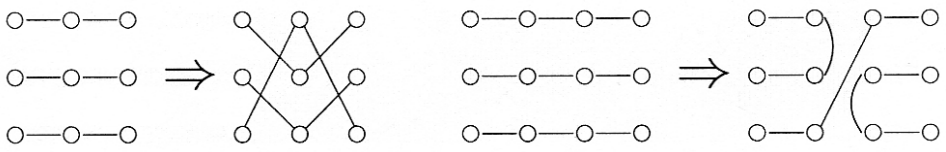

Fig. 4. Two large size neighborhoods.

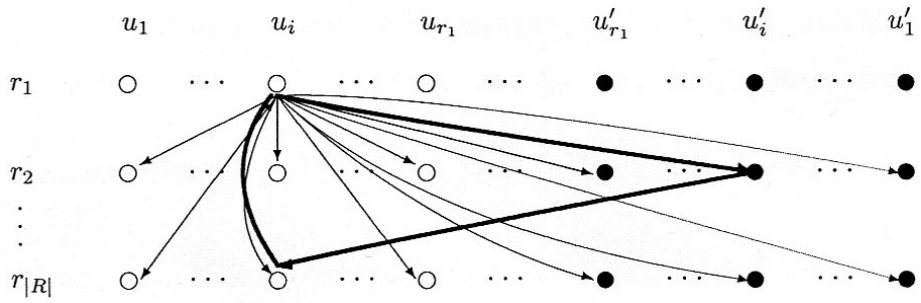

Fig. 5. Splitting and concatenation of some routes.

traversed both in the forward and backward direction; therefore, for every vertex u_i, we introduce its "double" u'_i and (for the illustrative purpose) represent the vertices in the layer first in the direct and then in the reverse order.

Let $u_i \in r$ and $v_j \in \overline{r}$. Now the interlayer arcs will be of the four types: (u_i, v_j), (u_i, v'_j), (u'_i, v_j), and (u'_i, v'_j).

- Arc (u_i, v_j) means the construction of the route $(u_1, \ldots, u_i, v_{j+1}, \ldots, v_{\overline{r}})$, its weight $w(u_i, v_j)$ equals $c_{u_i v_{j+1}} - c_{u_i u_{i+1}}$.
- Arc (u_i, v'_j) corresponds to the creation of the route $(u_1, \ldots, u_i, v_j, \ldots, v_1)$, its weight $w(u_i, v'_j)$ is equal to $c_{u_i v_j} - c_{u_i u_{i+1}}$.
- Arc (u'_i, v_j) signifies making of the route $(u_r, \ldots, u_{i+1}, v_{j+1}, \ldots, v_{\overline{r}})$, its weight $w(u'_i, v_j)$ equals $c_{u_{i+1} v_{j+1}} - c_{u_i u_{i+1}}$.
- Arc (u'_i, v'_j) means the creation of the route $(u_r, \ldots, u_{i+1}, v_j, \ldots, v_1)$, its weight $w(u'_i, v'_j)$ is equal to $c_{u_{i+1} v_j} - c_{u_i u_{i+1}}$.

If the construction of a new route leads to exceeding the vehicle capacity then we set the weight of the arc equal to a sufficiently large number. The presence of a cycle in this graph that passes through each layer at most once and has negative weight indicates the possibility of restructuring the solution with decrease of the objective function value. The search for such a cycle is implemented in the same way as in the previous case.

In Fig. 5 the cycle in bold corresponds to restructuring of the three routes: the first, second, and last. The beginning of the first route is concatenated to the start of the second; the end of the second route, to the end of the last; and the beginning of the last, to the end of the first route. Some variant of such a concatenation is shown in Fig. 4.

4. COMPUTATIONAL EXPERIMENTS

The above-developed algorithms are implemented in Java language and tested on the PC with the processor Intel©Core^{TM} i7, 2.20 GHz, 8 GB of RAM. In the first experiment, we tested the effectiveness of the subgradient optimization method to solve problem (9)–(12) for a given sequence of customers. For this purpose, the test instances were used from [12] where the number of customers varied from 19 to 255.

Table 2. The results for the given sequences

n	< 100	100–150	150–200	> 200	19–255
Number of instances	15	38	31	12	96
$\mathrm{Iter}_{\mathrm{avg}}$	4	8	18	8	11
Err_{avg}	0.06	0.43	0.33	0.33	0.33
$\mathrm{LbE}_{\mathrm{avg}}$	0.12	0.5	0.68	0.33	0.48
t_{avg}	1.24	4.45	13.74	20.83	8.99
T	186	1690	4259	2500	8635
No solutions found	0	0	0	0	0

Table 3. The results for mixing inside the routes

n	< 100	100–150	150–200	> 200	19–255
Number of instances	15	38	31	12	96
$\mathrm{Iter}_{\mathrm{avg}}$	31	68	110	37	72
Err_{avg}	−4.39	−3.89	−5.69	−5.92	−4.8
$\mathrm{LbE}_{\mathrm{avg}}$	1.73	2.89	2.47	1.87	2.45
t_{avg}	3.25	16.03	58.56	50.7	32.1
T	488	6093	18155	6084	308204
No solutions found	1	3	6	0	10

The same paper contains the record solutions for those instances. These solutions can be represented as a sequence of customers just as it is done in the intensification process. For the subgradient optimization method the following parameters were used:

$$\alpha = 0.74, \qquad \Theta^0 = 1.8, \qquad UB = 1.3LR(\lambda^t).$$

The value Θ^i was divided by 1.1 after every four iterations. Each instance was solved 10 times. The method stopped in the case of finding a first feasible solution or after exhausting the maximum number of iterations. We calculated the average number of iterations for obtaining the first feasible solution $\mathrm{Iter}_{\mathrm{avg}}$, the average error (in percentage terms) relative to the lower estimate $\mathrm{LbE}_{\mathrm{avg}}$, the average running time t_{avg}, and the total running time T of the algorithm in milliseconds. The maximum number of iterations did not exceed 2000.

Table 2 presents the results of calculations in five columns: the examples with

$$n < 100, \qquad 100 \le n < 150, \qquad 150 \le n < 200, \qquad n \ge 200,$$

together with the summarizing column for all examples with $19 \le n \le 255$. The line $\mathrm{Err}_{\mathrm{avg}}$ shows the average error (in percentage terms) of the obtained solution relative to the record one.

In this experiment, for 4 examples out of 96 we improved the record solution. In 54 cases the algorithm found a record solution. In other cases it lost, but the average error turned out to be small, only 0.33%. Since the comparison was performed for certain well-known test cases, where many algorithms were tested, the application of the subgradient optimization methods can be considered as reasonable.

Table 4. The results for random sequences

n	< 100	100–150	150–200	> 200	19–255
Number of instances	15	38	31	12	96
$\mathrm{Iter}_{\mathrm{avg}}$	134	218	249	138	205
$\mathrm{LbE}_{\mathrm{avg}}$	6.94	3.82	3.08	2.77	3.94
t_{avg}	8.37	40.15	91.42	110.79	60.57
T	1256	15258	28340	13295	58149
No solutions found	6	15	16	2	39

As it was noted above, in case of an unlucky choice of sequence the algorithm may fail to find an feasible solution. The problem can just have no solutions. Therefore, the experiment was performed for other sequences for the same examples. In the first case, a randomly ordered sequence of customers was selected for each route. In the second case, an random sequence among all customers was taken. Note that, in the first case, a feasible solution always exists.

Tables 3 and 4 present the results of this experiment. The value of Err_{avg} in Table 3 demonstrates the average relative error of the final solution in comparison with the initial one. In the first case, in only 10 tests out of 960 we failed to find a feasible solution. For random sequence, this number turned out to be large, but was equal only to 39; i.e., about 4% of the total number of tests. The average error did not exceed 4%, although, for small number of customers, it was almost two times higher, more precisely, 6.94%. We can conclude that for these examples we rarely fail to find a feasible solution of the problem. Thus, restrictions (4) are not rigid.

In the second experiment, we tested the effectiveness of the hybrid algorithm HLS. For these purposes, we used the same test instances. The algorithm parameters were $I = 100$ and $I_{LS} = n$. In Perturb procedure, the move to the next neighboring solution was performed $\lceil 3I/I_{LS} \rceil$ times.

The developed algorithm was compared with the two heuristics that are the most effective among the availables: GRASP-ELS, an adaptive greedy heuristic with evolutionary local search [12], and MS-ILS-SFR, a hybrid scheme of iterative local search with multistart and Simple fleet reassignment procedure [16]. The first heuristic was tested on the PC with the Opteron processor, 2.1 GHz; the second heuristic, on the PC with the processor Intel©Core^{TM} i7 2.9 GHz, 8GB RAM. All experiments were performed using a single processor core.

The results of numerical experiments are presented in Tables 5–8. The first column "Code" gives the name of the example by which we can easily reconstruct the characteristics of engaged vehicles. The second column contains the dimension n of the problem.

For the HLS algorithm there are specified the minimum and average values $z_{\min}$ and z_{avg}, standard deviation σ, and the average computation time t in seconds for 10 runs of the algorithm. For other algorithms, we present the value of the objective function z and the computation time t in seconds. The penultimate column contains the value z^* of the objective function of the best known solution; and the last column, the relative error g (in percentage terms) of the mean value z_{avg} against the available record. The new records that are obtained by the hybrid algorithm HLS are highlighted in bold.

As a whole, the combination of the methods of subgradient optimization, local search, and new procedures for intensification and diversification has demonstrated its effectiveness. For fifteen test instances we improved the record values of the objective function. The average relative error over all test cases was 0.65%.

Table 5. The results of computational experiments for $n < 150$

		HLS				MS-ILS-SFR		GRASPxELS			
Code	n	$z_{\min}$	z_{avg}	σ	t	z	t	z	t	z^*	g
H-75	19	452.85	452.85	0.00	1	452.85	1	452.85	0	452.85	0.00
H-92	34	564.39	564.39	0.00	29	564.39	21	564.39	21	564.39	0.00
H-93	39	1036.99	1036.99	0.00	19	1036.99	17	1036.99	27	1036.99	0.00
H-94	46	1378.25	1378.25	0.00	45	1378.25	27	1378.66	16	1378.25	0.00
H-55	55	10256.16	10264.37	10.20	46	10244.34	28	10247.86	191	10244.34	0.20
H-52	58	4035.59	4035.59	0.00	61	4027.27	63	4029.42	40	4027.27	0.21
H-10	68	2108.10	2108.32	0.50	185	2107.55	119	2107.55	25	2107.55	0.04
H-39	76	2926.59	2928.99	2.88	202	2921.36	171	2934.55	182	2921.36	0.26
H-70	77	6689.61	6729.87	19.77	201	6684.56	125	6689.61	121	6684.56	0.68
H-82	78	4769.35	4772.58	1.87	180	4766.74	108	4774.26	145	4766.74	0.12
H-08	83	4596.52	4597.65	1.14	254	4591.75	93	4598.49	305	4591.75	0.13
H-36	84	5709.31	5738.34	24.22	358	5684.62	267	5759.34	104	5684.61	0.95
H-43	85	8737.13	8749.47	9.08	496	8707.94	178	8764.75	220	8707.94	0.48
H-01	91	**9180.45**	9187.28	5.93	387	9210.14	269	9210.14	52	9210.14	-0.25
H-11	94	3367.41	3372.57	2.86	430	3367.41	244	3370.47	265	3367.41	0.15
H-90	101	2350.68	2358.22	3.55	454	2347.50	332	2360.83	15	2346.13	0.52
H-17	104	5381.19	5403.10	15.82	439	5369.31	248	5370.05	173	5362.83	0.75
H-84	104	7228.38	7244.86	12.77	677	7236.49	344	7269.55	206	7227.88	0.23
H-81	105	**10675.92**	10690.71	9.57	546	10689.77	385	10715.28	84	10689.77	0.01
H-2B	106	8577.50	8601.68	17.30	859	8482.79	538	8537.31	303	8464.69	1.62
H-07	107	**8074.64**	8135.00	27.90	562	8089.21	245	8130.50	306	8089.21	0.57

CONCLUSION

We considered the vehicle routing problem in the case of heterogeneous limited vehicle fleet. To solve the problem, we applied an approach in which we split the problem into two subproblems: the search for the best sequence of customers and the optimal distribution of the vehicle fleet for a given sequence of customers. For the first subproblem we applied the method of local search with variable neighborhoods; and for the second subproblem, the subgradient optimization method.

We incorporated the new procedures of intensification and diversification into the scheme of local search, as well as the procedure of post-optimization based on the neighborhoods of exponential size.

The computational experiments demonstrated the effectiveness of the proposed approach. For 15 test instances we improved the record values of the objective function.

Table 6. The results of computational experiments for $n < 150$ (continuation of Table 5)

		HLS				MS-ILS-SFR		GRASPxELS			
Code	n	$z_{\min}$	z_{avg}	σ	t	z	t	z	t	z^*	g
H-87	107	3753.87	3757.12	4.60	511	3753.87	432	3753.87	104	3753.87	0.09
H-47	110	16291.49	16332.46	32.13	598	16175.22	391	16222.94	334	16175.22	0.97
H-48	110	**21316.55**	21444.07	58.62	663	21330.75	520	21413.92	371	21318.04	0.59
H-61	110	7308.84	7320.97	6.46	559	7292.03	519	7300.10	108	7292.03	0.40
H-12	111	3543.99	3545.82	1.87	632	3543.99	536	3543.99	71	3543.99	0.05
H-30	111	6329.09	6361.97	13.81	550	6322.39	425	6321.69	201	6313.39	0.77
H-2A	112	7820.37	7862.82	27.45	1200	7793.16	448	7885.93	299	7793.16	0.89
H-53	114	6483.51	6497.73	12.37	663	6434.83	511	6470.49	418	6434.83	0.98
H-05	115	10896.35	10937.73	31.76	898	10896.33	267	10963.62	489	10896.33	0.38
H-13	118	6709.28	6728.61	11.94	854	6696.43	368	6713.14	303	6696.43	0.48
H-06	120	11760.08	11783.45	15.76	1453	11711.35	389	11792.94	368	11692.85	0.77
H-03	123	**10727.36**	10747.53	14.35	1433	10730.10	673	11320.58	512	10730.10	0.16
H-83	123	10041.06	10050.45	7.75	1339	10029.60	635	10019.83	332	10019.15	0.31
H-74	124	11598.92	11634.25	27.60	1247	11592.72	536	11732.54	247	11586.87	0.41
H-21	125	5160.03	5175,71	13.23	835	5139.84	408	5154.38	330	5139.84	0.70
H-26	125	**6393.47**	6433.33	22.57	1071	6423.70	1045	6481.93	351	6423.70	0.15
H-88	126	12406.93	12452.74	17.95	1236	12448.38	410	12443.41	632	12402.85	0.40
H-16	128	4156.97	4160.43	2.26	1081	4156.97	799	4161.61	181	4156.97	0.08
H-31	130	4105.67	4130.97	15.92	2052	4091.81	1174	4103.88	308	4091.52	0.96
H-40	131	11156.86	11184.18	14.16	1401	11122.32	808	11172.98	615	11122.32	0.56
H-89	133	**7099.68**	7120.97	14.34	1331	7105.47	682	7135.36	246	7105.47	0.22
H-41	134	7606.16	7643.93	17.54	1915	7572.07	969	7679.32	326	7572.07	0.95
H-34	135	5784.25	5799.40	9.19	1565	5786.98	784	5800.12	406	5758.09	0.72
H-60	136	17073.80	17106.54	17.25	1262	17037.23	647	17067.85	444	17037.23	0.41
H-73	136	**10195.13**	10215.26	10.05	1636	10196.04	536	10243.66	598	10195.33	0.20
H-28	140	5538.45	5550.86	6.67	1560	5533.01	525	5542.76	343	5531.06	0.36
H-25	142	7209.61	7218.87	6.12	2438	7217.26	1722	7228.54	518	7206.64	0.17
H-85	145	8812.03	8842.44	16.86	2257	8797.92	787	8874.31	383	8779.76	0.71
H-79	146	7266.75	7310.23	16.36	1687	7274.18	1138	7314.89	474	7259.54	0.70
H-66	149	**12790.56**	12862.79	49.36	1978	12828.34	1316	13319.73	443	12828.34	0.27

Table 7. The results of computational experiments for $150 \leq n < 200$

		HLS				MS-ILS-SFR		GRASPxELS			
Code	n	$z_{\min}$	z_{avg}	σ	t	z	t	z	t	z^*	g
H-69	151	9222.25	9270.15	32.43	2130	9167.18	940	9276.93	509	9162.78	1.17
H-76	151	**12007.57**	12036.84	20.71	4265	12018.22	1241	12098.66	686	12018.22	0.15
H-56	152	31236.61	31286.03	42.21	2254	31090.53	1136	31292.81	394	31090.53	0.63
H-86	152	9056.31	9085.76	15.13	2512	9053.41	908	9076.63	383	9030.68	0.61
H-37	160	6894.98	6915.59	18.16	2468	6870.11	1553	6921.19	384	6858.23	0.84
H-64	160	17157.37	17180.65	21.47	1984	17135.16	792	17157.37	512	17135.16	0.27
H-24	162	9145.18	9178.51	23.99	3357	9118.01	1722	9183.78	610	9101.47	0.85
H-57	162	**43378.37**	43581.83	112.20	2883	44850.05	1361	45152.42	639	44818.18	-2.76
H-29	163	9151.41	9182.21	16.88	2207	9142.86	1363	9147.39	425	9142.86	0.43
H-09	166	7671.00	7700.14	20.40	3126	7651.33	877	7647.59	450	7619.19	1.06
H-35	167	9605.62	9640.89	26.18	3274	9592.43	1163	9640.80	459	9574.71	0.69
H-45	169	10664.81	10719.12	38.93	2518	10496.88	2118	10519.25	451	10484.23	2.24
H-80	170	6833.02	6844.95	9.06	2864	6825.46	1530	6839.96	230	6816.89	0.41
H-44	171	12549.34	12604.56	32.95	2863	12237.42	1852	12418.00	447	12197.46	3.34
H-54	171	10433.38	10477.30	37.21	3380	10370.09	2725	11511.62	364	10370.09	1.03
H-67	171	10971.29	11024.31	36.86	3547	10915.60	1495	11854.61	337	10915.60	1.00
H-63	173	20154.56	20262.46	55.19	3686	19994.01	984	20241.72	694	19951.76	1.56
H-14	175	5680.64	5705.25	14.04	3138	5667.82	2498	5679.80	449	5644.92	1.07
H-42	177	10940.03	11012.21	38.18	5513	10855.73	3696	11713.90	317	10855.73	1.44
H-02	180	11797.23	11839.34	16.63	4987	11718.86	1693	12102.01	326	11718.86	1.03
H-04	182	10861.29	10892.04	20.09	4237	10787.03	1709	11276.45	726	10787.03	0.97
H-95	183	6223.54	6255.38	22.29	1877	6175.62	907	6244.13	322	6175.62	1.29
H-71	185	9976.24	9988.03	11.32	3417	9891.50	1294	9960.84	640	9870.22	1.19
H-72	185	5950.67	5977.67	17.57	3647	5883.33	2238	5976.54	197	5883.33	1.60
H-50	186	12466.27	12482.40	15.01	5488	12385.32	4092	12508.77	647	12374.04	0.88
H-15	187	8255.95	8273.64	10.00	5706	8268.18	2224	8301.63	521	8236.40	0.45
H-33	188	**9419.00**	9452.46	25.23	6089	9437.30	2277	9543.17	603	9421.01	0.33
H-77	189	6943.61	7036.46	35.32	3852	6929.67	2804	6991.59	636	6929.67	1.54
H-78	189	7101.20	7162.22	22.33	4247	7039.90	1937	7069.82	471	7035.01	1.81
H-59	192	14299.28	14324.68	18.66	5405	14309.48	2974	14367.14	676	14282.59	0.29
H-91	195	**6374.01**	6398.88	16.00	4918	6381.13	3235	6437.14	544	6377.48	0.34

Table 8. The results of computational experiments for $n > 200$

		HLS				MS-ILS-SFR		GRASPxELS			
Code	n	$z_{\min}$	z_{avg}	σ	t	z	t	z	t	z^*	g
H-23	202	7769.33	7793.84	14.89	5154	7760.62	2657	7809.20	802	7750.27	0.56
H-38	204	11224.72	11271.90	24.97	6177	11217.53	2613	11439.58	422	11217.53	0.48
H-27	219	**8417.62**	8444.78	21.38	7536	8436.55	3424	8520.74	996	8436.55	0.10
H-58	219	23480.52	23543.71	40.30	8724	23504.15	2640	23530.10	1028	23397.76	0.62
H-65	222	13036.33	13102.95	28.69	8043	13013.89	4348	13077.63	636	13013.89	0.68
H-19	223	11760.02	11819.28	28.73	6831	11702.98	2588	11805.34	1010	11702.77	1.00
H-62	224	23035.91	23094.21	49.26	8710	22952.06	3220	23434.56	829	22952.06	0.62
H-22	238	13240.93	13270.10	15.43	8341	13103.51	2424	13162.90	836	13068.03	1.55
H-32	243	**9378.30**	9418.40	22.65	16911	9412.56	5771	9537.48	1131	9412.56	0.06
H-49	245	16282.44	16368.02	54.37	18026	16219.41	8694	16417.30	990	16219.41	0.92
H-46	249	24697.34	24832.07	84.31	11514	24428.54	7371	24805.27	1475	24428.54	1.65
H-18	255	9676.56	9703.47	17.31	13305	9668.17	6127	9797.61	1216	9668.17	0.37

ACKNOWLEDGMENTS

The authors were supported by the Russian Humanitarian Science Foundation (project no. 13–22–10002).

REFERENCES

1. I. A. Davydov, P. A. Kononova, and Yu. A. Kochetov, "Local Search with an Exponential Neighborhood for the Servers Load Balancing Problem," Diskretn. Anal. Issled. Oper. **21** (6), 21–34 (2014) [J. Appl. Indust. Math. **9** (1), 27–35 (2015)].
2. I. A. Davydov, Yu. A. Kochetov, N. Mladenović, and D. Urošević, "Fast Metaheuristics for the Discrete $(r|p)$-Centroid Problem," Avtomat. i Telemekh. No. 4, 106–119 (2014) [Automat. Remote Control **75** (4), 677–687 (2014)].
3. P. A. Kononova and Yu. A. Kochetov, "The Variable Neighborhood Search for the Two Machine Flow Shop Problem with a Passive Prefetch," Diskretn. Anal. Issled. Oper. **19** (5), 63–82 (2012) [J. Appl. Indust. Math. **7** (1), 54–67 (2013)].
4. P. I. Stetsyuk, *Ellipsoid Methods and r-Algorithms* (Evrika, Chişinău, 2014) [in Russian].
5. R. Baldacci, M. Battarra, and D. Vigo, "Routing a Heterogeneous Fleet of Vehicles," in *The Vehicle Routing Problem: Latest Advances and New Challenges*, Ed. by B. Golden, S. Raghavan, and E. Wasil (Springer, New York, 2008), pp. 3–27.
6. R. Baldacci and A. Mingozzi, "A Unified Exact Method for Solving Different Classes of Vehicle Routing Problems," Math. Program. Ser. A **120** (2), 347–380 (2009).
7. M. Boudia, C. Prins, and M. Reghioui, "An Effective Memetic Algorithm with Population Management for the Split Delivery Vehicle Routing Problem," in *Hybrid Metaheuristics*: *Proceedings of the 4th International Workshop on Hybrid Metaheuristics (Dortmund, Germany, Oct. 8–9, 2007)* (Springer, Berlin, 2007), pp. 16–30.
8. J. Brandão, "A Deterministic Tabu Search Algorithm for the Fleet Size and Mix Vehicle Routing Problem," European J. Oper. Res. **195** (3), 716–728 (2009).
9. J. Brandão, "A Tabu Search Algorithm for the Heterogeneous Fixed Fleet Vehicle Routing Problem," Comput. Oper. Res. **38** (1), 140–151 (2011).

10. J. Desrosiers, F. Soumis, M. Desrochers, and M. Sauvé, "Methods for Routing with Time Windows," European J. Oper. Res. **23** (2), 236–245 (1986).
11. C. Duhamel, C. Gouinaud, P. Lacomme, and C. Prodhon, "A Multi-Thread GRASPxELS for the Heterogeneous Capacitated Vehicle Routing Problem," in *Hybrid Metaheuristics*, Ed. by El-G. Talbi (Springer, Heidelberg, 2013), pp. 237–269,
12. C. Duhamel, P. Lacomme, and C. Prodhon, "A GRASPxELS with depth First Search Split Procedure for the HVRP," Res. Rep. LIMOS/RR-10-08 (Inst. Supér. Inform., Modél. Appl., Aubière, France, 2010). Available at http://www.isima.fr/lacomme/doc/RR_HVRP1-4_V1.pdf.
13. C. Duhamel, P. Lacomme, and C. Prodhon, "Efficient Frameworks for Greedy Split and New Depth First Search Split Procedures for Routing Problems," Comput. Oper. Res. **38** (4), 723–739 (2011).
14. F. Li, B. Golden, and E. Wasil, "A Record-to-Record Travel Algorithm for Solving the Heterogeneous Fleet Vehicle Routing Problem," Comput. Oper. Res. **34** (9), 2734–2742 (2007).
15. P. H. V. Penna, A. Subramanian, and L. S. Ochi, "An Iterated Local Search Heuristic for the Heterogeneous Fleet Vehicle Routing Problem," J. Heuristics **19** (2), 201–232 (2013).
16. P. H. V. Penna, T. Vidal, L. S. Ochi, and C. Prins, "New Compound Neighborhoods Structures for the Heterogeneous Fixed Fleet Vehicle Routing Problem," in *Proceedings of the XLV Symposium Bras. Pesqui. Operac. (Natal, Brazil, September 16–19, 2013)*, (Soc. Bras. Pesqui. Oper., Rio de Janeiro, 2013), pp. 3623–3633 [Available at http://ws2.din.uem.br/ademir/sbpo/sbpo2013/pdf/arq0110.pdf].
17. B. T. Poljak, "Subgradient Methods: A Survey of Soviet Research," in *Nonsmooth Optimization*: *Proceedings of IIASA Workshop (Laxenburg, Austria, March 28–April 8, 1977)*, Ed. by C. Lemarechal and R. Mifflin (Pergamon Press, Oxford, GB, 1977), pp. 5–29.
18. J.-Y. Potvin and M.-A. Naud, "Tabu Search with Ejection Chains for the Vehicle Routing Problem with Private Fleet and Common Carrier," J. Oper. Res. Soc. **62** (2), 326–336 (2011).
19. C. Prins, "Efficient Heuristics for the Heterogeneous Fleet Multitrip VRP with Application to a Large-Scale Real Case," J. Math. Model. Algorithms **1** (2), 135–150 (2002).
20. C. Prins, "Two Memetic Algorithms for Heterogeneous Fleet Vehicle Routing Problems," Eng. Appl. Artif. Intell. **22** (6), 916–928 (2009).
21. A. Subramanian, P. H. V. Penna, E. Uchoa, and L. S. Ochi, "A Hybrid Algorithm for the Heterogenous Fleet Vehicle Routing Problem," European J. Oper. Res. **221** (2), 285–295 (2012).
22. E. D. Taillard, "A Heuristic Column Generation Method for the Heterogeneous Fleet VRP," RAIRO, Oper. Res. **33** (1), 1–14 (1999).
23. C. D. Tarantilis, C. T. Kiranoudis, and V. S. Vassiliadis, "A List Based Threshold Accepting Metaheuristic for the Heterogeneous Fixed Fleet Vehicle Routing Problem," J. Oper. Res. Soc. **54** (1), 65–71 (2003).
24. C. D. Tarantilis, C. T. Kiranoudis, and V. S. Vassiliadis, "A Threshold Accepting Metaheuristic for the Heterogeneous Fixed Fleet Vehicle Routing Problem," European J. Oper. Res. **152** (1), 148–158 (2004).
25. C. D. Tarantilis, E. E. Zachariadis, and C. T. Kiranoudis, "A Guided Tabu Search for the Heterogeneous Vehicle Routing Problem," J. Oper. Res. Soc. **59** (12), 1659–1673 (2008).
26. T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "A Unified Solution Framework for Multi-Attribute Vehicle Routing Problems," European J. Oper. Res. **234** (3), 658–673 (2014).