TRANSPORT
ISSN 1648-4142 print / ISSN 1648-3480 online
2013 Volume 28(1): 46–59
doi:10.3846/16484142.2013.781540

METAHEURISTIC APPROACH TO TRANSPORTATION SCHEDULING IN EMERGENCY SITUATIONS

Peter Korošec[1], Gregor Papa[2]

Computer Systems Dept of, Jožef Stefan Institute
Jamova cesta 39, SI-1000 Ljubljana, Slovenia
E-mails: [1]peter.korosec@ijs.si; [2]gregor.papa@ijs.si (corresponding author)

Submitted 5 July 2011; accepted 10 October 2011

Abstract. This paper compares two metaheuristic approaches in solving a constrained transportation scheduling problem, which can be found in transporting goods in emergency situations. We compared Greedy Search, Parameter-less Evolutionary Search and Ant-Stigmergy Algorithm. The transportation scheduling/allocation problem is NP-hard, and is applicable to different real-life situations with high frequency of loading and unloading operations; like in depots, warehouses and ports. To evaluate the efficiency of the presented approaches, they were tested with four tasks based on realistic data. Each task was evaluated using group and free transportation approach. The experiments proved that all tested algorithms are viable option in solving such scheduling problems, however some performing better than others on some tasks.

Keywords: transportation, emergency situation, vehicle scheduling, metaheuristic optimization, genetic algorithm, greedy search, ant-colony optimization.

Introduction

One type of transportation is the movement of goods from one location to another. It can be performed by different modes, such as air, rail, road and water. Its efficiency depends on infrastructure, vehicles, and operations. The growth of the transportation, especially road transport, increases congestion and environmental pollution, and decreases safety (Sivilevičius 2011), therefore it should be performed as optimal as possible.

In emergency situations we are often limited to road transportation, since air, rail or water transportation are either unavailable, damaged, or slow. Since usually the infrastructure cannot be optimized ad-hoc, the efficiency can be improved mainly by the optimization of vehicle fleet and the operations sequence/schedule. Despite numerous publications on efficient scheduling methods for vehicle routing, very few address the loading schedule with multiple constraints; on different types of cargo, different vehicle capacities, different loading bay capacities, and overall transportation time minimization.

In this paper we describe a special version of vehicle routing problem (VRP) (Dantzig, Ramser 1959; Bertsimas, Van Ryzin 1991; Toth, Vigo 2001; Zeng *et al.* 2007; Lin *et al.* 2009), where the vehicles circulate between two depots only (Salhi, Petch 2007). The stress is on optimal schedule of vehicles and allocations of vehicles to loading bays for loading and unloading of cargo, at the depots, and can be treated as job shop scheduling problem (Beck *et al.* 2003). Here, each vehicle can be represented as a resource and each visit to loading/unloading bay with a selected cargo type as an activity. Such transformation proves that our problem is also NP-hard (Garey, Johnson 1979).

The transportation scheduling case is especially useful in emergency/rescue situations (like floods, earthquakes, etc.), when large amounts of goods have to be transported from one depot to another as quickly as possible, with the limited number of available vehicles. Like VRP, this is also a combinatorial optimization problem (Lenstra, Rinnooy Kan 1981). The emergency/rescue situations and disaster relief situations are covered in number of papers (Doerner, Hartl 2008; Özdamar, Yi 2008); some of them also solve the problem with genetic algorithm (GA) (Bae *et al.* 2007; Salhi, Petch 2007; Saadatseresht *et al.* 2009) and ant-colony optimization (Tatomir, Rothkrantz 2006; Rizzoli *et al.* 2007; Yi, Kumar 2007; Lee *et al.* 2010).

For solving scheduling problems, many scheduling methods are reported in the literature (Chiong, Dhakal 2009; Guo *et al.* 2009; Jarboui *et al.* 2009; Wang, Tang 2009; Xing *et al.* 2010). One of the nature-inspired approaches (Chiong 2009) is the GA. A heuristic for the open job shop scheduling problem using the GA to minimize makespan was developed by (Senthilkumar, Shahabudeen 2006). On the other hand, a scheduling method based on the GA was developed by considering multiple criteria in (Chryssolouris, Subramaniam 2001).

To solve the loading schedule problem we used two metaheuristic techniques that have already proven as efficient methods for solving combinatorial optimization problems. The first approach was Parameter-Less Evolutionary Search (PLES) (Papa 2008), and the second one was Ant-Stigmergy Algorithm (ASA) (Korošec, Šilc 2008). The PLES was used in the process of the search for an optimal production plan (Papa *et al.* 2012), and in the process of geometry optimization of an electrical motor stator and rotor (Papa 2008). The ASA was used in the process of geometry optimization of an electrical motor stator and rotor, geometry optimization of radial impeller, and geometry optimization of electrical motor case (Oblak *et al.* 2007; Tušar *et al.* 2007).

In comparison to the previous paper (Papa, Korošec 2009), this paper deals with more transportation tasks. These tasks differ in number of vehicles, capacities of loading bays, as well as in amount of cargo to be transported. The tasks used in this paper are based on the possible scenarios of Civil Protection, however the approach is also useful when selecting logistic centers locations (Turskis, Zavadskas 2010). Furthermore, in this paper the graph construction process of the ASA was changed, to achieve better performance in dealing with constraints. Special care was given to the constraints regarding capabilities of loading different cargo types at each loading bay. Also, the PLES encoding was slightly improved to achieve faster convergence regardless of the task.

The rest of the paper is organized as follows: in Section 1 we formally define the problem; in Section 2 we describe approaches used for the search of the best schedule; in Section 3 we present the evaluation procedure; in Section 4 we describe the experimental environment; in Section 5 we show the experimental results; and in end of paper we list some conclusions and possible future work.

1. Definitions

A problem includes v vehicles, and c types of cargo (i.e., container, box, barrel, pallet...), with quantities V_j, where $j = 1, ..., c$. For each vehicle we know the capacity for cargos w_{ij}, where $i = 1, ..., v$ and $j = 1, ..., c$. The capacity w_{ij} is measured in a cargo-relevant unit, i.e., ton, cubic meter, number of containers, etc. Not all cargo types are suitable for each vehicle. If the vehicle i is not capable of loading type j of a cargo, then its capacity is $w_{ij} = 0$.

Each vehicle can load in each run one type of a cargo only, however in different runs the same vehicle can load different cargos. The vehicle is always fully loaded; unless the remaining cargo is smaller than the capacity of the vehicle.

For each vehicle the maximum speed is known (for loaded and empty vehicle). With the distance between depot A and B we can calculate the driving times $t_{l1}, ..., t_{lv}$ from A to B for loaded vehicles, and $t_{e1}, ..., t_{ev}$ from B to A for empty vehicles.

At depot A there are m loading bays, each capable of loading a_{zj} of cargo j per hour, $z = 1, ..., m$. If the loading bay is not appropriate for type j of the cargo, then $a_{zj} = 0$. Similarly, at depot B there are n unloading bays, capable of unloading b_{uj} of cargo j per hour, $u = 1, ..., n$. There can be only one vehicle at a time on each loading or unloading bay.

A vehicle with self-loader is capable of loading a_{0j} of cargo j per hour, and unloading b_{0j} of cargo j per hour. The vehicle with a self-loader does not have to wait in a queue for the empty loading/unloading bay. It can be loaded/unloaded either at the appropriate and available loading bay, or outside the loading bay (e.g., somewhere at the warehouse/depot area).

The aim is to transport different types of cargo from depot A to B. Since the cargo cannot be transported in one run, vehicles have to come back from B to A and perform several runs. Each run starts in A and consists of:

- waiting in queue till assigned loading bay is available;
- loading at loading bay;
- driving from A to B;
- waiting in queue till assigned unloading bay is available;
- unloading at unloading bay;
- driving from B to A.

The scheduling operation consists of assigning a type of a cargo for each vehicle, a loading bay and an unloading bay. The aim is to find a schedule of cargo types and loading/unloading bays for all vehicles to finish the transportation of all cargos in the shortest possible time. The general approach for solving this problem is represented in Algorithm 1, while the whole transportation process for each vehicle is presented in Fig. 1.

The evaluation of each proposed schedule was done using a problem-specific simulator. The simulator followed the schedule and considered all the constraints, defined with the problem.

Algorithm 1. An approach to transportation problem solving:

```
while all cargo not transported do
    for all vehicles do
        if current vehicle will be used then
            Choose among available cargo types for this vehicle.
            Reduce cargo according to vehicle's capacity.
            Choose among possible loading bays for a chosen cargo type.
            Choose among possible unloading bays for a chosen cargo type.
        end if
    end for
end while
```

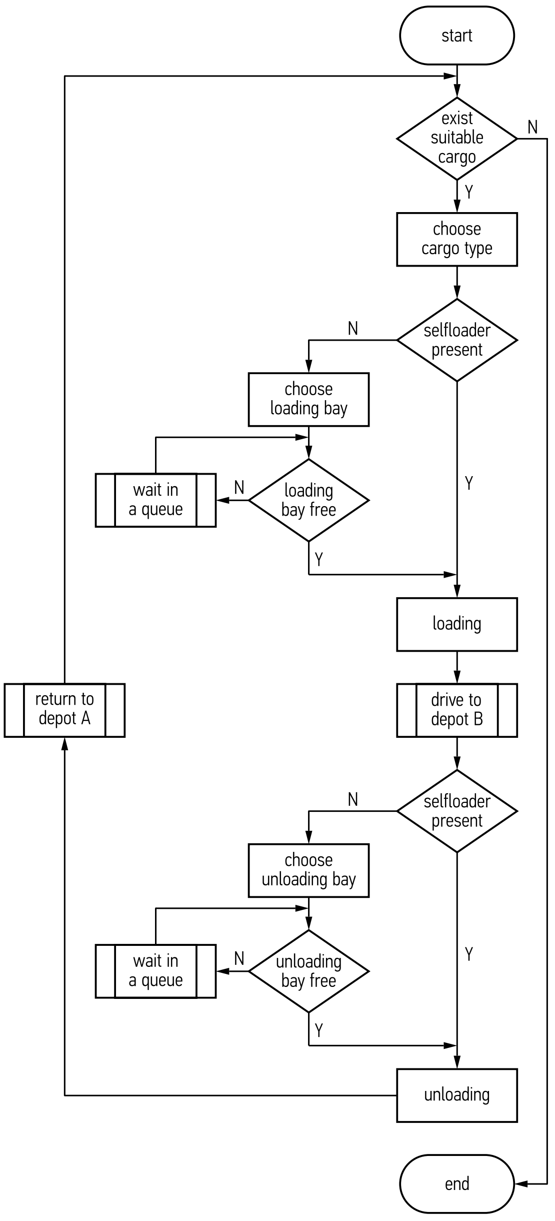

Fig. 1. Transportation process for each vehicle

1.1. Problem Representation

In our case, each vehicle run is represented with a number of parameters. The first parameter represents the type of cargo and a special value that denotes idle run (i.e., this vehicle run is omitted from schedule), followed by a number of parameters in pairs, which represent possible loading and unloading bay for each cargo type represented in the first parameter. So, if we had two cargo types defined in the first parameter, then we would need four additional parameters which will represent possible loading and unloading bays for both types. Due to such presentation, only one pair of parameters is used when the solution is being evaluated according to the selected cargo type and all others are being omitted. So, for r runs and v vehicles we need $k = r\sum_{i=1}^{v}(1+2l_i)$ values, where l_i is the number of possible cargos for vehicle i.

1.2. Problem Variations

In this paper, we will be solving two variations of the basic problem. The first variation, called the group transportation, has a constraint that all vehicles must travel in a convoy formation. This means, that no vehicle can leave the depot until all vehicles are fully (un)loaded and the traveling speed of a convoy equals the speed of the slowest vehicle. The second variation, called the free transportation, has no such constraints. This means, that a vehicle leaves the depot as soon as it is (un)loaded and it travels at its own speed. Both scenarios are possible and the choice mainly depends on the characteristics of the emergency situation. If safety is the main concern then convoy is the preferred way, if speed is more important, then a free transportation is a more suitable option as it will be shown in Section 5.

2. Algorithms

2.1. The Parameter-Less Evolutionary Search

The PLES (Papa 2008) is based on a basic GA (Goldberg 1989), except that it does not need any control parameter, e.g., population size, number of generations, probabilities of crossover and mutation, to be set in advance. They are set virtually, according to the size of the problem and according to the statistical properties of the solutions found. In its search process the PLES tries to efficiently explore the whole search space in order to find the optimal solution (see Algorithm 2).

Algorithm 2. Parameter-Less Evolutionary Search:

```
SetInitialPopulation(P)
Evaluate(P)
Statistics(P)
while not EndingCondition() do
    ForceBetterSolutions(P)
    MoveSolutions(P)
    Evaluate(P)
    Statistics(P)
end while
```

Population Initialization and Termination Criterion

The vehicle schedule was encoded into one chromosome with k genes, as presented in Section 2.1. The initial population P consists of $PopSize_0$ chromosomes. In each chromosome the distribution of cargo type and loading/unloading bays are randomly distributed. The initial population size is set according to the following equation:

$$PopSize_0 = 4v\sqrt{k} + 10\lg(\textstyle\sum_{i=1}^{k} combinations_i),$$

where: $combinations_i$ is the number of possible combinations of the i-th variable (either the number of possible cargo types or the number of possible loading/unloading bays).

The EndingCondition() function checks to see if there was no improvement for several generations; then the system is assumed to be in the steady state, and the optimization is then ended. The number of generations depends on the convergence speed of the best solution found. The optimization keeps running while a better solution is found every few generations. But when there is no improvement of the best solution for a few generations (*Limit*), the optimization process stops. It depends on the current size of the population:

$$Limit = 10\lg(PopSize).$$

It is proportional to the population size, where larger populations are used to quickly exploit the search space to find some near-optimal regions, while smaller populations are used to explore this region more precisely. The rough search of the large population should have enough time, while the fine-tuning within the optimal regions should not take too many generations without improvement.

Force a Better Solution

In every generation the worse solutions are replaced with better solutions, and up to 25% of the genes in the chromosomes are switched. This operator incorporates the function of elitism, while forcing the replacement of worse solutions with better ones, and the function of crossover, while taking the good solutions and slightly changing them at some positions.

Solution Moving

Typically, every chromosome is the subject of a mutation in the basic GA. In the PLES, the mutation is realized by moving some positions in the chromosome according to the population's statistical properties. Namely, the positions are not mutated to some random value, but are moved in a direction towards the best chromosome in the population.

First, only the solutions that were not changed within the 'Force better' operator are handled here. In other words, the solutions of the previous generation that were better than the average are changed. The number of positions in the chromosome (*Ratio*) to be moved is calculated on the basis of the standard deviation of the solutions in the previous generation and the maximum standard deviation as stated in the following equation:

$$Ratio_i = \frac{StDev_{i-1}}{StDev_{i-3}} k,$$

where: $StDev_{i-1}$ and $StDev_{i-3}$ are the standard deviation of the solution fitness of the previous generation, and the standard deviation three generations ago, respectively. Here, the $Ratio \in [0, k]$, and the *Ratio* positions in the chromosome are selected to be moved. When the standard deviation of the current population is high, i.e., the chromosomes are distributed far from the optimal regions, the number of mutated positions in the chromosome is high so as to allow larger jumps to other, probably better, regions in the search space. When the standard deviation is low, only a few positions are mutated to allow for a finer search inside the regions that are close to the optimal solution. The moves are implemented by changes of the cargo type and/or by changes of (un)loading bays.

Solution Evaluation and Statistics

Each population is statistically evaluated. Here the best, the worst, and the average fitness value in the generation are found. Furthermore, the standard deviation of the fitness values of all solutions in the generation, the maximal standard deviation of the fitness value over all the generations, and the average value of each parameter in the solution are calculated.

2.2. The Ant-Stigmergy Algorithm

The ASA (Korošec, Šilc 2008) is an approach to solving multi-parameter optimization problems and it is based on stigmergy, a type of collective work that can be observed in ant colonies. It roots can be found in the ant-colony optimization (ACO) metaheuristic first proposed by Dorigo and colleagues (Dorigo 1992; Dorigo *et al.* 1999). The basic concept of the algorithm is as follows: first, the multi-parameter problem is translated into a search graph and then an optimization technique is used to find the cheapest path in the constructed graph; this path consists of the values of the optimized parameters.

Search Graph Construction

The search graph construction (*SearchGraphConstruction*) consists of the translation of the discrete parameter values of the problem into a search graph $G = (V, E)$ with a set of vertices, V, and a set of edges, E, between the vertices. For each parameter x_d, $d = 1, \ldots, D$ the set of parameter values, $V_d = \{v_{d,1}, \ldots, v_{d,n_d}\}$, is constructed. For all vertices $v_{d,i}$, $i = 1, \ldots, n_d$, the path length from the *start* vertex to any of the vertices is equal to d, and d is also called the vertex *distance* in graph G. Here, n_d depends on the number of values that are possible for the d-th parameter. A vertex $v_{d,i}$ represents one vertex in a search graph, and each vertex of the parameter x_d is connected to all the vertices that belong to the next parameter x_{d+1} (see Fig. 2). In this way, the constrained

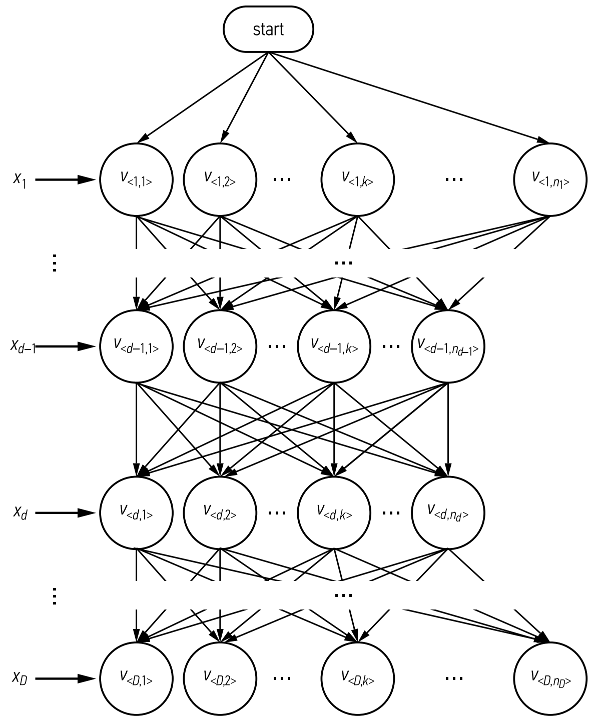

Fig. 2. Search graph representation

multi-parameter optimization problem is transformed into a problem of finding the cheapest path.

Optimization

Optimization consists of finding the cheapest path. Prior to the actual optimization an initial amount of pheromone, τ^0, is deposited equally on all the vertices in the search graph (*SearchGraphInitialization*). There are a number of ants in a colony, all of which begin simultaneously from the *start* vertex. The probability with which they choose the next vertex depends on the amount of pheromone on the vertices. Ants use a probability rule to determine which vertex will be chosen next. More specifically, ant in step d moves from vertex $v_{d-1,i} \in \{v_{d-1,i}, ..., v_{d-1,n_{d-1}}\}$ to vertex $v_{d,j} \in \{v_{d,1}, ..., v_{d,n_d}\}$ with the probability given by:

$$p_{ij,\alpha}(d) = \frac{\tau_{d,j}}{\sum_{1 \le k \le n_d} \tau_{d,j}},$$

where: $\tau_{d,k}$ is the amount of pheromone on vertex $v_{d,k}$. The ants repeat this action until they reach the ending vertex (*FindPath*). Then, the gathered parameter values of each ant (which can be found on its path) are evaluated (*Evaluate*). Next, each ant returns to the *start* vertex and on the way it deposits pheromone on the vertices according to the evaluation result: the better the result the more pheromone is deposited on the vertices, and vice versa (*UpdatePheromone*). After all the ants have returned to the start vertex a so-called daemon action is made, which in this case consists of depositing some additional pheromone on what is currently the best path (*DaemonAction*). Afterwards, pheromone evaporation from all the vertices occurs (*EvaporatePheromone*), i.e. the amount of pheromone is decreased by some predetermined percentage on each vertex $v_{d,k}$ in the search graph G:

$$\tau_{d,k} \leftarrow (1-p)\tau_{d,k}.$$

The whole procedure is then repeated until some ending condition is met. The outline of the ASA pseudo code is shown in Algorithm 3.

Algorithm 3. Ant-Stigmergy Algorithm:

```
graph = SearchGraphConstruction(parameters)
SearchGraphInitialization(initial pheromone amount)
while not current level ending condition do
    for all ants in colony do
        path = FindPath(graph)
        Evaluate(path)
    end for
    UpdatePheromone (all found paths vertices)
    DaemonAction(best path)
    EvaporatePheromone(all vertices)
end while
```

3. Evaluation

Table 1 presents an example of short schedule produced by one of the algorithms. The first column presents the vehicle to be used; the second column presents the cargo type; while the third and the fourth column present the loading and unloading bay, respectively. Loading/unloading bay marked as 0 denotes the usage of the self-loader outside the loading bay.

Since we are dealing with two different variations, the schedule shown in Table 1 has also two different evaluation 'executions'.

Table 1. Example of the vehicles' schedule and loading bay allocation

Vehicle	Cargo type	Loading bay	Unloading bay
1	3	3	3
2	1	4	4
3	1	4	1
5	2	2	2
7	2	1	0
9	2	1	4
10	3	4	2
13	1	0	1
17	3	2	3
19	2	1	2
20	3	5	4
2	3	3	3
5	2	5	2
6	1	4	4
7	2	1	4
9	2	1	4
12	2	5	0
14	1	0	0
17	3	2	3
20	3	5	4

3.1. Group Transportation

At group transportation, we look at all vehicles and their corresponding scheduled run. At first run, we notice that only vehicles 1, 2, 3, 5, 7, 9, 10, 13, 17, 19, and 20 will be used to transport cargo (Fig. 3a), while all the other vehicles stay at the depot. Since vehicle 3 is loading from the same bay as vehicle 2, it must first wait until vehicle 2 is loaded, before it can be loaded. So, at any time only one vehicle can be loaded at one bay. The time from the start of loading to convoy release from the depot equals the time needed for the last vehicle to be loaded with cargo. Only then the convoy can start their trip to destination depot B (Fig. 3b). The time needed for a convoy to reach destination equals the time of the slowest vehicle. The time of unloading is determined by the time that is needed for all vehicles to unload the cargo. Here also only one vehicle can be unloaded on one bay (Fig. 3c). Only after then, the convoy can return to starting depot A with a speed of the slowest vehicle (Fig. 3d). When the vehicles reach the depot the whole procedure is repeated for all subsequent runs until all cargo is transported (Fig. 3e).

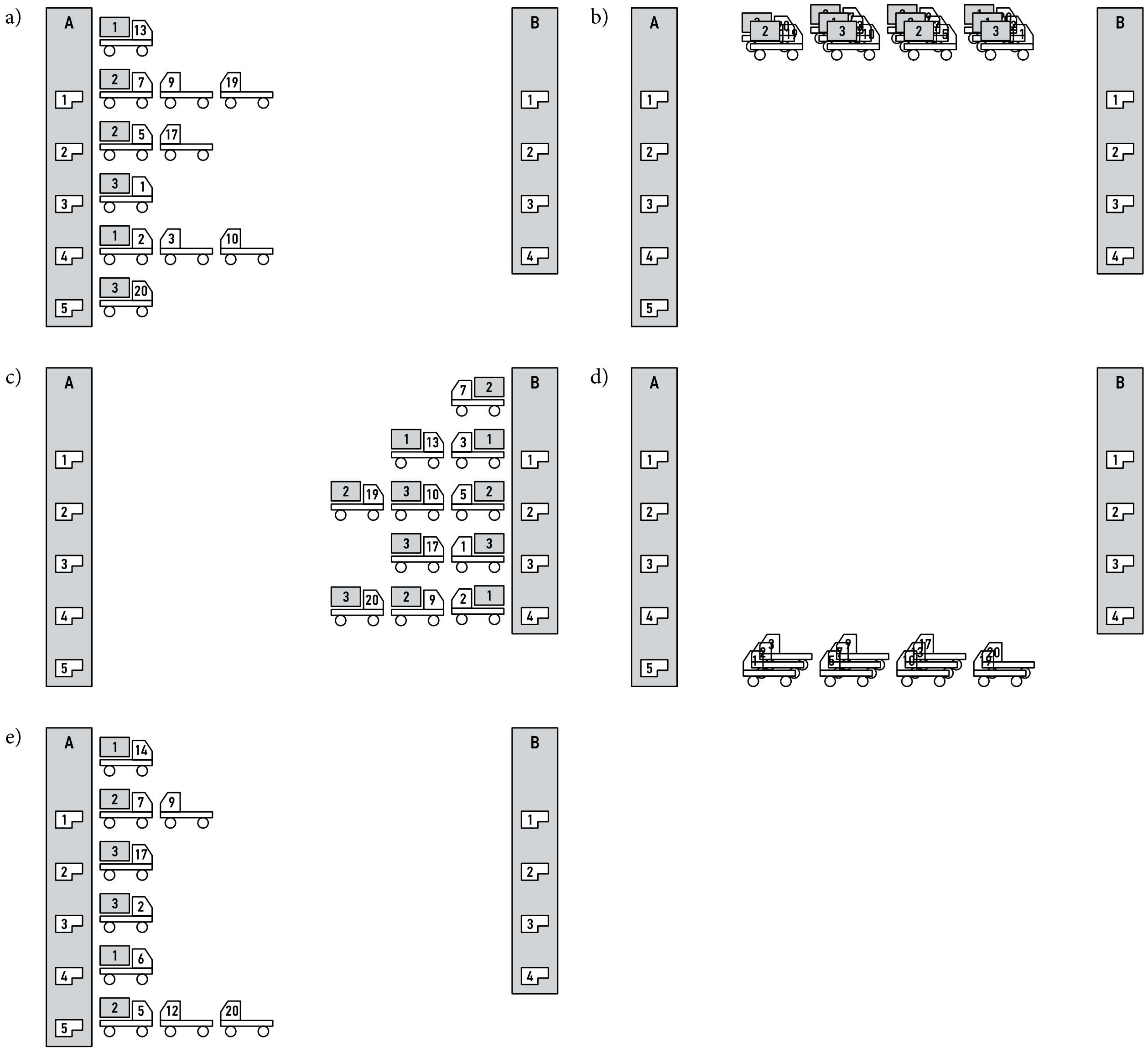

Fig. 3. Group transportation example: a – initial loading at *A*; b – moving of a group of vehicles from *A* to *B*; c – unloading at the *B*; d – returning of empty vehicles to *A*; e – loading of the next group of vehicles

3.2. Free Transportation

At free transportation, we look at the schedule on vehicle per vehicle basis. So the first vehicle that is scheduled is vehicle 1. Since it is the first, it is put on bay 3. Since next vehicle (in this example it is number 2) is scheduled for different bay than vehicle 1, it can be also put there at the same time as vehicle 1. On the other hand, vehicle 3 is put in a queue of bay 4, since vehicle 2 is already there. It must wait until the cargo is loaded on vehicle 2, before being put for loading. This continues until we come to the next scheduled run of vehicle 2 (line 12 of example schedule in Table 1). Since vehicle 2 has not finished previously planned run, we wait with this schedule until vehicle 2 comes back from destination depot *B*. But this does not stop us to check if there exists any other vehicle, further down the schedule, that is not on route. In this example, we can see that vehicle 6 is not on the route yet, so it is put on a queue of bay 4. In Fig. 4a, we see the situation at the depot *A* after all vehicles are placed in scheduled queues. The vehicle leaves for destination depot *B* at the very moment it is loaded with cargo. So, vehicle does not wait for other vehicles like it was the case at group transportation (Fig. 4b). The time needed to get to the destination depot *B* depends only on its own speed. As like with the loading, it must wait for a free bay and only then unload the cargo (Fig. 4c). After unloading, it does not wait for other vehicles, but goes back immediately (Fig. 4d). When it returns, it is put on the bay that is scheduled next. In case of vehicle 5, this would be bay 5 (Fig. 4e). This procedure is being repeated until all cargo is transported to destination depot *B*.

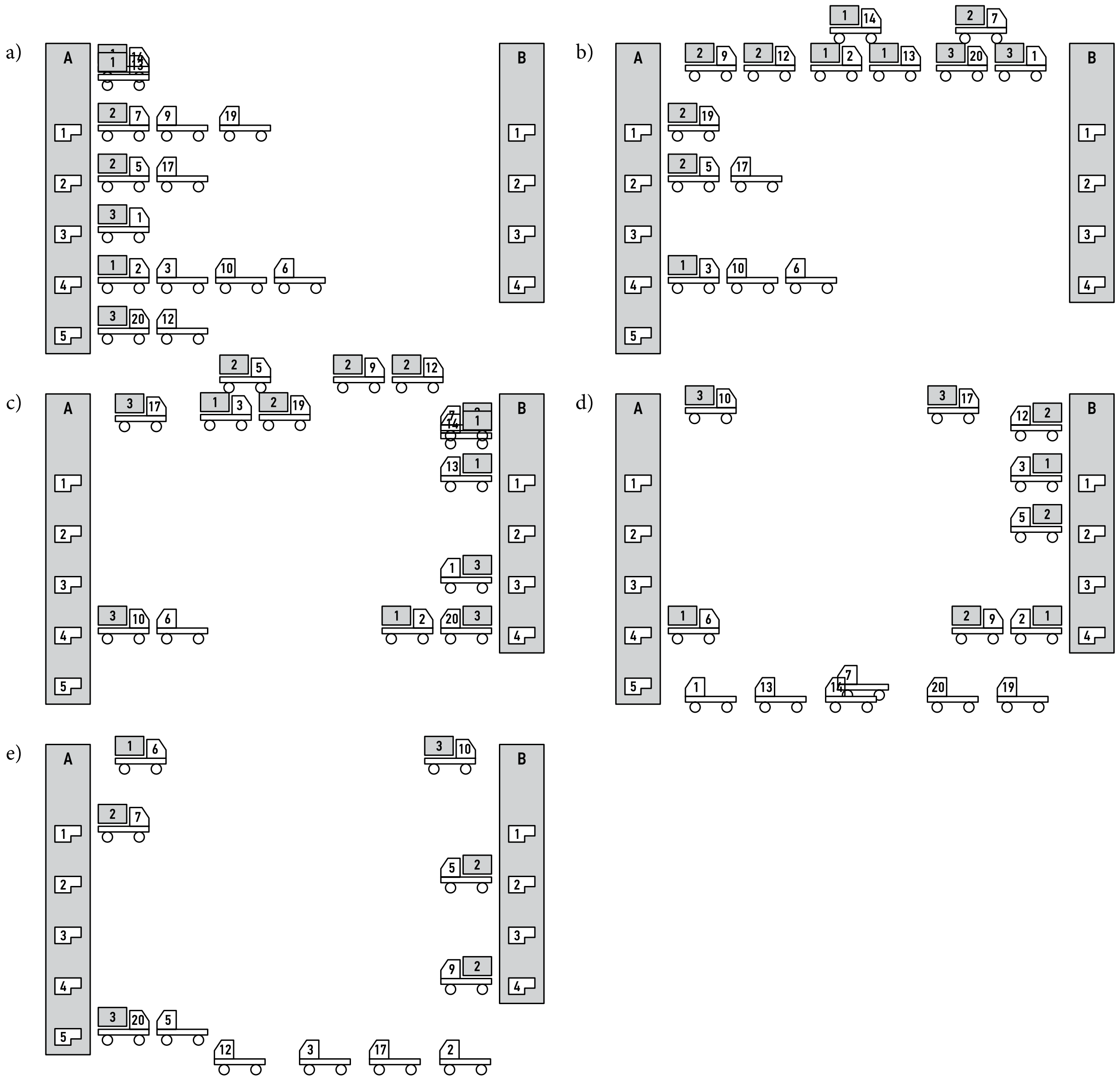

Fig. 4. Free transportation example: a – initial loading at *A*; b – moving of loaded vehicles from *A* to *B* and loading of queued vehicles; c – unloading of the vehicles at *B*; d – returning of the empty vehicles to *A* and unloading of the remaining at *B*; e – loading of the of vehicles in the second run

4. Experimental Environment

The computer platform used to perform the experiments was based on Intel i7 3.5-GHz processor, 12 GB of RAM running Windows 7 x64 operating system. The PLES was implemented in Microsoft Visual C++, while the ASA was implemented in Borland Delphi. The main reason for usage of different program languages is in a fact that the basic algorithms were already implemented in those programming languages. Therefore, we did not have to do any time-consuming recoding.

4.1. Task Description

The experiments presented in this paper consist of four tasks. These tasks differ in properties of the loading bays, in the number of vehicles, and in the quantity of cargos. In all tasks for each loading/unloading bay the capacities vary for different cargo types. The tasks were made on the basis of possible scenarios that might be used by national Civil Protection. This includes sizes of various national civil-protection depots, as well as their distance from several larger cities. The number of vehicles is set by a possible number of vehicles located at some depots, and also some additional vehicles that might be enrolled in case of emergency. Since the vehicles and personnel have to gather at some depot, there is some time available to run the program to produce the optimal schedule of the vehicles for the given situation (distance, amount of cargo, etc.). So, considering that time, the scheduler was constrained to give a result in a period of 5÷10 minutes.

Table 2. Cargo quantities to be transported

Tasks 1, 2, and 3			
Cargo type	1	2	3
Quantity	30	1000	200000
Task 4			
Cargo type	1	2	3
Quantity	35	1800	200

Table 3. Vehicle capacities for different cargo types

Tasks 1, 3, and 4					
Number of vehicles	2	4	6	4	4
Type 1 capacity	2	2	–	1	–
Type 2 capacity	38	50	12	8	8
Type 3 capacity	15000	20000	5000	–	3000
Self-loader	no	no	yes	yes	no
Task 2					
Number of vehicles	1	1	2		
Type 1 capacity	2	2	–		
Type 2 capacity	38	50	8		
Type 3 capacity	–	20000	10		
Self-loader	no	yes	yes		

Table 4. Loading/unloading bays capacities for different cargo types

Tasks 1, 2, and 4				
Bay	Bay number	Capacity		
		Type 1	Type 2	Type 3
Loading	1	4	30	5000
Loading	2	–	50	5000
Loading	3	–	–	40000
Loading	4	4	35	4000
Loading	5	–	40	7000
Unloading	1	4	30	10000
Unloading	2	–	50	5000
Unloading	3	–	–	50000
Unloading	4	3	35	4000
Task 3				
Bay	Bay number	Capacity		
		Type 1	Type 2	Type 3
Loading	1	4	30	5000
Loading	2	–	50	50
Loading	3	–	–	40000
Loading	4	40	3	4000
Loading	5	–	40	7000
Loading	6	–	40	–
Loading	7	67	–	–
Loading	8	45	40	7000
Loading	9	–	–	7000
Unloading	1	10	30	10
Unloading	2	–	50	5000

For all experiments we had three types of a cargo. In Task 1 there were: 20 vehicles (consisting of five different types of vehicles); five loading bays and 4 unloading bays. In Task 2 there were only four vehicles (three different types). In Task 3 there were nine loading bays and only 2 unloading bays. In Task 4 the cargo capacities were changed. Table 2, Table 3, and Table 4 present the detailed properties for each Task. The vehicles differ in different cargo capacities as well as they had different speeds. Also, not all vehicles were able to transport every type of cargo and some vehicles also had self-loaders. The capacities of loading bays at depot *A* and unloading bays at depot *B* are different. Self-loading capacity was defined by the vehicles, and was always lower compared to the loading bay capacity.

For all four Tasks the distance between depots *A* and *B* was 90 kilometers, with three equally long sections, each having the different highest possible speed (80, 130, and 90 km/h). Indeed, the real speed of each vehicle depends on their capacities and on regarding whether they were full (on their way from *A* to *B*) or empty (returning from *B* to *A*).

4.2. Experiments

Two algorithms (PLES and ASA) were used to find the shortest time for a task of transportation of cargos from depot *A* to *B*. The greedy search approach (Plestenjak *et al.* 2008) was used for acquiring comparative results presented in Table 5 based on arranging vehicles according to their efficiency of transporting certain types of cargos. According to approach (Plestenjak *et al.* 2008), it would be ideal if one could use the same amount of time to transfer each of the cargo types. To attain this goal, at a loading bay such a cargo was chosen for each vehicle at each run, so that the cargo volume ratio V_1:V_2:V_3 of transported cargo was maintained the same throughout the whole transportation process. And at depot *B*, the unloading bay was chosen to unload the cargo in the shortest time possible.

Table 5. Greedy search results for each Task

Task	Result	
	Group	Free
1	32.25	18.56
2	79.99	76.41
3	68.63	43.34
4	53.79	24.18

We determined the lower bound of the number of runs for all vehicles. The lower bound was set according to the capacity of all cargos and capacities of all vehicles for each cargo. It is calculated as follows:

$$r^* = \frac{V_1}{\sum_{i=1}^{v} w_{i1}} + \frac{V_2}{\sum_{i=1}^{v} w_{i2}} + \frac{V_3}{\sum_{i=1}^{v} w_{i3}}.$$

This lower bound was used to determine the upper bound, which was used to narrow the search space and decrease the search time. The upper bound used for the

PLES and ASA was set to $r = 2r^*$. The reason for using the upper bound was due to the fact that some vehicles can be omitted in some runs, therefore in general more runs are needed and only few solutions are feasible with $r = r^*$. With r set to some higher value the algorithm is able to find quickly more feasible solutions and then successfully converge to lower values, which are usually close to r^*.

Since the PLES is parameter-less algorithm, there was no need to set any control parameters. Stopping criteria was automatically set according to the convergence progress, i.e., approximately 30000 evaluations or when maximal number of evaluations is reached, i.e., 500000 evaluations. The ASA parameters were set as follows: the number of ants was 200, $\rho = 0.001$ and stopping criteria was determined by maximal number of evaluations, 500000, or 50000 evaluations without improvement. To set these parameters several test executions of the algorithm were done.

To obtain final solution the algorithms needed approximately 200000 to 500000 evaluations, which in time took up to 10 minutes. Regardless of the algorithm, to transport the cargos, all vehicles had to perform about 7 runs in Task 1, 33 runs in Task 2, and 11 runs in Tasks 3 and 4. Each algorithm was executed 20 times.

5. Results

5.1. Group Transportation

Tables 6–9, corresponding to Tasks 1–4, respectively, present the best, worst, mean, and standard deviation value of the group transportation time (measured in hours). The fifth row presents standard error of the mean (SEM).

Table 6. Performance of the PLES and the ASA for group transportation in Task 1

	PLES	ASA
Best	37.04	26.69
Worst	50.05	33.01
Mean	43.84	30.79
St. dev.	3.69	1.63
SEM	0.83	0.37
Median	44.32	31.32
Conf. int. 99%	±2.13	±0.94
Avg. evaluations	500000	418320

Table 7. Performance of the PLES and the ASA for group transportation in Task 2

	PLES	ASA
Best	85.83	76.42
Worst	99.66	80.97
Mean	91.64	78.47
St. dev.	4.01	1.73
SEM	0.90	0.39
Median	90.96	77.91
Conf. int. 99%	±2.31	±1.00
Avg. evaluations	290719	290930

Table 8. Performance of the PLES and the ASA for group transportation in Task 3

	PLES	ASA
Best	79.93	56.84
Worst	105.31	70.12
Mean	95.96	63.76
St. dev.	6.83	4.13
SEM	1.53	0.92
Median	97.35	65.24
Conf. int. 99%	±3.93	±2.38
Avg. evaluations	500000	402030

Table 9. Performance of the PLES and the ASA for group transportation in Task 4

	PLES	ASA
Best	46.09	41.82
Worst	52.41	46.95
Mean	49.61	44.25
St. dev.	1.66	1.47
SEM	0.37	0.33
Median	49.57	44.23
Conf. int. 99%	±0.96	±0.84
Avg. evaluations	490253	426210

The sixth row presents the median. The seventh row presents the confidence interval for the mean value with the confidence range of 99%. The last row presents the average number of evaluations needed to finish the algorithm run.

Table 10 presents a set of non-parametric statistical tests (Wilcoxon 1945; Mann, Whitney 1947; Brandt 1976; Press *et al.* 1992; Demšar 2006) for both algorithms for group transportation in all Tasks. The level of F-test shows that the results of both algorithms can be treated as having equal variance in all four Tasks. According to T-test for two samples with equal variance the difference between results of PLES and ASA algorithms is extremely statistically significant for all Tasks, with the level of significance $\alpha = 0.01$. T-test requires normal distributions of samples, but since the size of the samples is relatively small (i.e., 20) the distribution shape could not be verified. Therefore, additional U-test was used to evaluate the results of a T-test. The results of U-test with $\alpha = 0.01$ are again smaller than 0.0001 for all Tasks, which additionally proves the statement above.

Table 10. Statistical test of the PLES and the ASA performance for group transportation in all Tasks

		Task 1	Task 2	Task 3	Task 4
F-test	p	0.0008	0.0006	0.0341	0.5966
T-test	$t(38)$	14.45	13.49	18.04	10.84
	p	0.0000	0.0000	0.0000	0.0000
U-test	z	5.42	5.42	5.42	5.29
	U	1	0	0	5
	p	< 0.0001	< 0.0001	< 0.0001	< 0.0001

Since we have found out that there exists statistically significant difference between compared algorithms, we must determine which one is better. So further statistical tests were performed, which enables us to determine just that. We used pairwise comparisons with the Wilcoxon signed-ranks test (Wilcoxon 1945) and the Bergmann–Hommel dynamic post-hoc procedure (Bergmann, Hommel 1988). Table 11 presents average rankings, while Tables 12 and 13 present pairwise comparisons with the Wilcoxon signed-ranks test and with the Bergmann–Hommel dynamic post-hoc procedure, respectively. Both tests confirm that the ASA returned significantly better solutions than the PLES in all four Tasks with the level of significance 0.01.

Fig. 5 presents the performance of both algorithms in all Tasks, where the average, minimal and maximal values are presented in the graph.

From results shown in this section we can conclude that the ASA performs significantly better than the PLES in all Tasks and has a smaller standard deviation. At the same time, the average results of both algorithms outperform greedy search in Task 4, while only the ASA outperforms greedy approach also on all remaining Tasks. This indicates that the ASA is a preferred approach in dealing with a group transportation scheduling problem.

5.2 Free Transportation

Similar to the above comparison, Tables 14–17, corresponding to Tasks 1–4 for free transportation approach, present the best, worst, mean, and standard deviation value of the transportation time (measured in hours). The fifth row presents SEM. The sixth row presents the median. The seventh row presents the confidence interval for the mean value with the confidence range of 99%. The last row presents the average number of evaluations needed to finish the algorithm run.

Table 11. Average rankings of the algorithms for group transportation

Algorithm	Task			
	1	2	3	4
PLES	2.0	2.0	2.0	2.0
ASA	1.0	1.0	1.0	1.0

Table 12. Pairwise comparisons with the Wilcoxon test for group transportation

Hypothesis	Task	R^+	R^-	p-value
ASA vs. PLES	1	210	0	< 0.0001
ASA vs. PLES	2	210	0	< 0.0001
ASA vs. PLES	3	210	0	< 0.0001
ASA vs. PLES	4	210	0	< 0.0001

Table 13. Multiple comparisons with the Bergmann–Hommel procedure for group transportation

Hypothesis	Task	Adjusted p-value
ASA vs. PLES	1	< 0.0001
ASA vs. PLES	2	< 0.0001
ASA vs. PLES	3	< 0.0001
ASA vs. PLES	4	< 0.0001

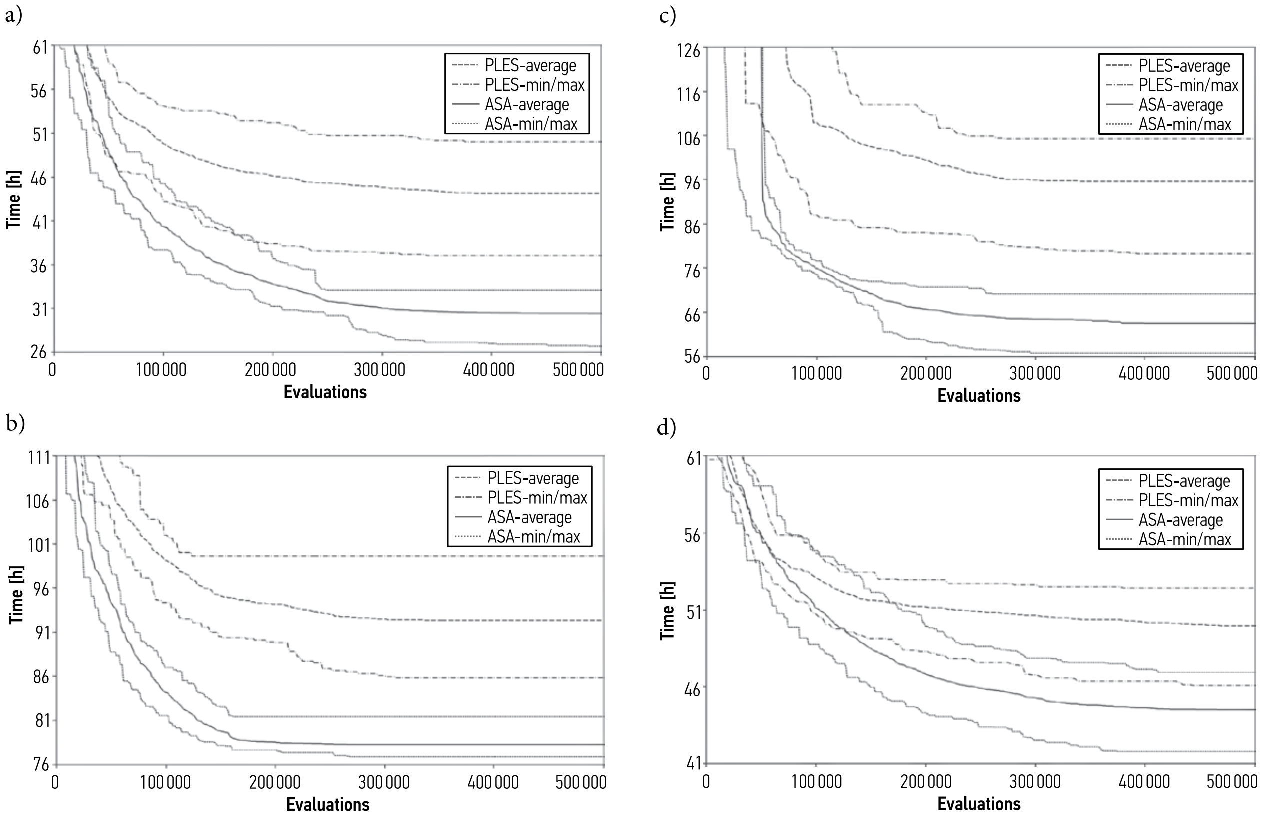

Fig. 5. Performance of the PLES and the ASA for group transportation in: a – Task 1; b – Task 2; c – Task 3; d – Task 4

Table 14. Performance of the PLES and the ASA for free transportation in Task 1

	PLES	ASA
Best	20.00	17.82
Worst	24.44	21.76
Mean	22.38	19.71
St. dev.	1.36	1.03
SEM	0.30	0.23
Median	22.51	19.50
Conf. int. 99%	±0.78	±0.59
Avg. evaluations	475305	331110

Table 15. Performance of the PLES and the ASA for free transportation in Task 2

	PLES	ASA
Best	66.17	65.05
Worst	74.85	68.33
Mean	70.59	66.36
St. dev.	2.25	1.18
SEM	0.50	0.26
Median	70.61	66.11
Conf. int. 99%	±1.29	±0.68
Avg. evaluations	433490	325420

Table 16. Performance of the PLES and the ASA for free transportation in Task 3

	PLES	ASA
Best	42.53	42.04
Worst	47.99	103.02
Mean	45.56	62.54
St. dev.	1.80	22.76
SEM	0.40	5.09
Median	45.29	62.58
Conf. int. 99%	±1.04	±13.11
Avg. evaluations	493332	380910

Table 17. Performance of the PLES and the ASA for free transportation in Task 4

	PLES	ASA
Best	25.07	23.77
Worst	27.07	27.01
Mean	25.93	25.09
St. dev.	0.59	0.79
SEM	0.13	0.18
Median	25.98	25.09
Conf. int. 99%	±0.34	±0.46
Avg. evaluations	369487	419620

Table 18 presents a set of non-parametric statistical tests for both algorithms in all Tasks for free transportation approach. The level of *F*-test shows that the results of both algorithms can be treated as having an equal variance in Tasks 1, 2 and 4. According to *T*-test for two samples with equal variance the difference between results of PLES and ASA algorithms is extremely statistically significant for Tasks 1, 2, and 4 with the level of significance $\alpha = 0.01$. The results of *U*-test with $\alpha = 0.01$ are again smaller than 0.001 for Tasks 1, 2, and 4, which additionally proves the statement above.

Table 19 presents average rankings, while Tables 20 and 21 present pairwise comparisons with the Wilcoxon signed-ranks test and with the Bergmann–Hommel dynamic post-hoc procedure, respectively. Both tests confirm that the ASA returned significantly better solutions than the PLES in Tasks 1, 2, and 4, while in Task 3 there is no significant difference between two algorithms with the level of significance 0.01.

Fig. 6 presents the performance of both algorithms in all Tasks, where the average, minimal and maximal values are presented in the graph.

From results shown in this section we can conclude that the ASA performs significantly better than the PLES in three Tasks and has only in two Tasks smaller standard deviation. The average results of both algorithms outperform greedy search only in Task 2. But neverthe-

Table 18. Statistical test of the PLES and the ASA performance for free transportation in all Tasks

		Task 1	Task 2	Task 3	Task 4
F-test	p	0.2315	0.0074	< 0.0001	0.2078
T-test	$t(38)$	7.04	7.46	3.33	3.79
	p	< 0.0001	< 0.0001	0.0020	0.0005
U-test	z	4.73	4.91	1.15	3.44
	U	25.5	19	158	73.5
	p	< 0.0001	< 0.0001	0.2501	0.0006

Table 19. Average rankings of the algorithms for free transportation

Algorithm	Task			
	1	2	3	4
PLES	2.0	2.0	1.4	2.0
ASA	1.0	1.0	1.6	1.0

Table 20. Pairwise comparisons with the Wilcoxon test for free transportation

Hypothesis	Task	R^+	R^-	p-value
ASA vs. PLES	1	210	0	< 0.0001
ASA vs. PLES	2	210	0	< 0.0001
ASA vs. PLES	3	36	174	0.01
ASA vs. PLES	4	210	0	< 0.0001

Table 21. Multiple comparisons with the Bergmann–Hommel procedure for free transportation

Hypothesis	Task	Adjusted p-value
ASA vs. PLES	1	< 0.0001
ASA vs. PLES	2	< 0.0001
ASA vs. PLES	3	0.371
ASA vs. PLES	4	< 0.0001

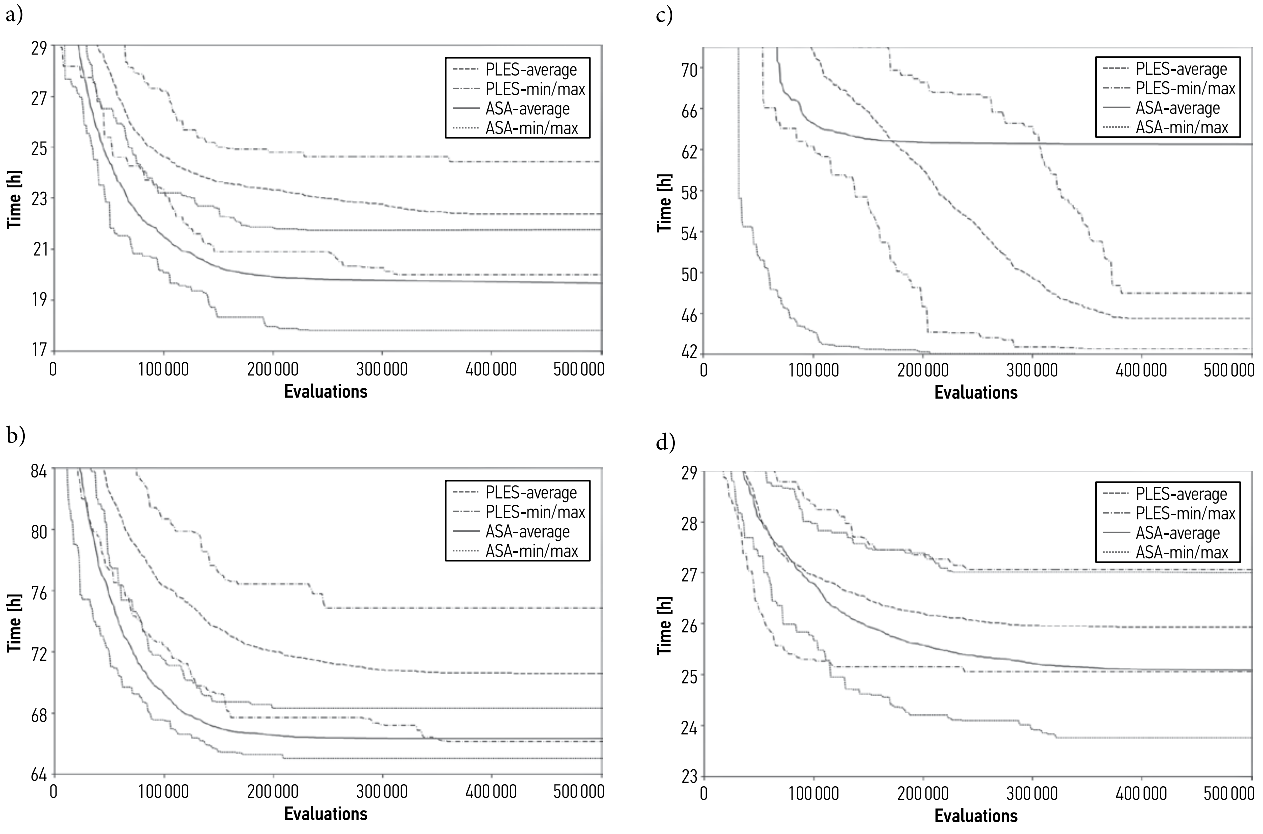

Fig. 6. Performance of the PLES and the ASA for free transportation in: a – Task 1; b – Task 2; c – Task 3; d – Task 4

less the best results obtained by the ASA are better than greedy search in all Tasks, while the PLES is better in Tasks 2 and 3. This indicates that if we have more time, the ASA is a preferred approach in dealing with a free transportation scheduling problem, but if this is not the case then greedy search is currently a better option.

Conclusion

The results show that the ASA found better solutions than the PLES in all Tasks except in Task 3 for free transportation approach. The main reason for lower performance in Task 3 is in "strong" local optima from which the ASA was not able to escape. But when it did, it still found better solutions then the PLES. Obviously, the constrained problems observed at the ASA graph construction process in (Papa, Korošec 2009) are with this new problem representation almost gone.

If we compare results obtained by all algorithms on group and free transportation approaches, we notice that on average greedy search performs better at free transportation, while it is much worse at group transportation. This indicates that the implemented version of greedy search is more adapted for solving free transportation. But since the ASA provides significantly better or even the best results than greedy search, a further investigation in providing more stable (smaller standard deviation) results is needed.

For further improvements to solutions of both algorithms a multi-restart approach is a viable possibility. This would be especially true for the ASA, since it could avoid strong local minima in Task 3 and decrease standard deviation. But this would further increase already high calculation time. So, next logical improvement would be to start from the solution obtained by the greedy search approach, since we would start with relatively high quality solution and with it decreased the calculation time. For the PLES the encoding should be further improved to achieve stable and better performance. Its poor behavior is seen as being prematurely stopped, by the number of evaluation limitation, in Tasks 1 and 3 of group transportation. Since its progress depends on the chromosome size, the improvement must be in the direction of chromosome encoding change.

In general, both algorithms performed well in all transportation cases, and could be easily used to solve this type of problems. The ASA was able to surpass the quality of solutions acquired by the greedy search approach in all instances, while the PLES in some of them. One advantage of the greedy search approach is its speed. To overcome this, an exploration of parallelization possibilities of the ASA and the PLES is planned in our future work. This is a sensible research direction since the number of cores per processor is increasing rapidly and not to mention the increasing popularity of GPU computing.

References

Bae, S. T.; Hwang, H. S.; Cho, G.-S.; Goan, M.-J. 2007. Integrated GA-VRP solver for multi-depot system, *Computers and Industrial Engineering* 53(2): 233–240. http://dx.doi.org/10.1016/j.cie.2007.06.014

Beck, J. C.; Prosser, P.; Selensky, E. 2003. Vehicle routing and job shop scheduling: what's the difference, in *Proceedings of the 13th International Conference on Artificial Intelligence Planning and Scheduling*, Trento, Italy, 267–276.

Bergmann, B.; Hommel, G. 1988. Improvements of general multiple test procedures for redundant systems of hypotheses, in P. Bauer, G. Hommel, and E. Sonnemann (Eds.). *Multiple Hypothesenprüfung-Multiple Hypotheses Testing*, 100–115.

Bertsimas, D. J.; Van Ryzin, G. 1991. A stochastic and dynamic vehicle routing problem in the Euclidean plane, *Operations Research* 39(4): 601–615. http://dx.doi.org/10.1287/opre.39.4.601

Brandt, S. 1976. *Statistical and Computational Methods in Data Analysis*. North Holland. 416 p.

Chryssolouris, G.; Subramaniam, V. 2001. Dynamic scheduling of manufacturing job shops using genetic algorithms, *Journal of Intelligent Manufacturing* 12(3): 281–293. http://dx.doi.org/10.1023/A:1011253011638

Chiong, R. (Ed.) 2009. *Nature-Inspired Algorithms for Optimisation*. Series: Studies in Computational Intelligence, Vol. 193. Springer. 536 p.

Chiong, R.; Dhakal, S. 2009. *Natural Intelligence for Scheduling, Planning and Packing Problems*. Springer. 352 p.

Dantzig, G. B.; Ramser, J. H. 1959. The truck dispatching problem, *Management Science* 6(1): 80–91. http://dx.doi.org/10.1287/mnsc.6.1.80

Demšar, J. 2006. Statistical comparisons of classifiers over multiple data sets, *Journal of Machine Learning Research* 7: 1–30.

Dorigo, M. 1992. *Ottimizzazione, apprendimento automatico ed algoritmi basati su metafora naturale*. Tesi di dottorato. Politecnico di Milano. 139 p. (in Italian).

Dorigo, M.; Di Caro, G.; Gambardella, L. M. 1999. Ant algorithms for discrete optimization, *Artificial Life* 5(2): 137–172. http://dx.doi.org/10.1162/106454699568728

Doerner, K. F.; Hartl, R. F. 2008. Health care logistics, emergency preparedness, and disaster relief: new challenges for routing problems with a focus on the austrian situation, in B. Golden *et al.* (Eds). *The Vehicle Routing Problem: Latest Advances and New Challenges*. 527–550. http://dx.doi.org/10.1007/978-0-387-77778-8_24

Garey, M. R.; Johnson, D. S. 1979. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. W. H. Freeman and Co. 340 p.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional. 432 p.

Guo, Y. W.; Li, W. D.; Mileham, A. R.; Owen, G. W. 2009. Applications of particle swarm optimisation in integrated process planning and scheduling, *Robotics and Computer-Integrated Manufacturing* 25(2): 280–288. http://dx.doi.org/10.1016/j.rcim.2007.12.002

Jarboui, B.; Eddaly, M.; Siarry, P. 2009. An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems, *Computers and Operations Research* 36(9): 2638–2646. http://dx.doi.org/10.1016/j.cor.2008.11.004

Korošec, P.; Šilc, J. 2008. The distributed multilevel ant-stigmergy algorithm used at the electric-motor design, *Engineering Applications of Artificial Intelligence* 21(6): 941–951. http://dx.doi.org/10.1016/j.engappai.2007.09.003

Lee, C.-Y.; Lee, Z.-J.; Lin, S.-W.; Ying, K.-C. 2010. An enhanced ant colony optimization (EACO) applied to capacitated vehicle routing problem, *Applied Intelligence* 32(1): 88–95. http://dx.doi.org/10.1007/s10489-008-0136-9

Lenstra, J. K.; Rinnooy Kan, A. H. G. 1981. Complexity of vehicle routing and scheduling problems, *Networks* 11(2): 221–227. http://dx.doi.org/10.1002/net.3230110211

Lin, S.-W.; Yu, V. F.; Chou, S.-Y. 2009. Solving the truck and trailer routing problem based on a simulated annealing heuristic, *Computers and Operations Research* 36(5): 1683–1692. http://dx.doi.org/10.1016/j.cor.2008.04.005

Mann, H. B.; Whitney, D. R. 1947. On a test of whether one of two random variables is stochastically larger than the other, *Annals of Mathematical Statistics* 18(1): 50–60. http://dx.doi.org/10.1214/aoms/1177730491

Oblak, K.; Korošec, P.; Kosel, F.; Šilc, J. 2007. Multi-parameter numerical optimization of selected thin-walled machine elements using a stigmergic optimization algorithm, *Thin-Walled Structures* 45(12): 991–1001. http://dx.doi.org/10.1016/j.tws.2007.07.006

Özdamar, L.; Yi, W. 2008. Greedy neighborhood search for disaster relief and evacuation logistics, *IEEE Intelligent Systems* 23(1): 14–23. http://dx.doi.org/10.1109/MIS.2008.7

Papa, G. 2008. Parameter-less evolutionary search, in *GECCO'08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*. Atlanta, GE. 1133–1134. http://dx.doi.org/10.1145/1389095.1389314

Papa, G.; Korošec, P. 2009. Metaheuristic approach to loading Schedule Problem, in *Proceedings of the Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2009),* 10–12 August 2009, Dublin, Ireland, 616–627.

Papa, G.; Vukašinović, V.; Korošec, P. 2012. Guided restarting local search for production planning, *Engineering Applications of Artificial Intelligence* 25(2): 242–253. http://dx.doi.org/10.1016/j.engappai.2011.07.001

Plestenjak, B.; Pisanski, T.; Dovgan, E.; Filipič, B.; Korošec, P.; Papa, G.; Medeot, M. 2008. Optimalni transport večje količine tovora med dvema lokacijama s skupino vozil, in *Proc. 9. slovenski kongres o cestah in prometu,* Portorož, Slovenia, 472–479 (in Slovenian).

Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, B. P. 1992. *Numerical Recipes in C: the Art of Scientific Computing*. Cambridge University Press. 994 p.

Rizzoli, A. E.; Montemanni, R.; Lucibello, E.; Gambardella, L. M. 2007. Ant colony optimization for real-world vehicle routing problems, *Swarm Intelligence* 1(2): 135–151. http://dx.doi.org/10.1007/s11721-007-0005-x

Saadatseresht, M.; Mansourian, A.; Taleai, M. 2009. Evacuation planning using multiobjective evolutionary optimization approach, *European Journal of Operational Research* 198(1): 305–314. http://dx.doi.org/10.1016/j.ejor.2008.07.032

Salhi, S.; Petch, R. 2007. A GA based heuristic for the vehicle routing problem with multiple trips, *Journal of Mathematical Modelling and Algorithms* 6(4): 591–613. http://dx.doi.org/10.1007/s10852-007-9069-2

Senthilkumar, P.; Shahabudeen, P. 2006. GA based heuristic for the open job shop scheduling problem, *The International Journal of Advanced Manufacturing Technology* 30(3–4): 297–301. http://dx.doi.org/10.1007/s00170-005-0057-2

Sivilevičius, H. 2011. Modelling the interaction of transport system elements, *Transport* 26(1): 20–34. http://dx.doi.org/10.3846/16484142.2011.560366

Tatomir, B.; Rothkrantz, L. 2006. Ant based mechanism for crisis response coordination, *Lecture Notes in Computer Science* 4150: 380–387. http://dx.doi.org/10.1007/11839088_36

Toth, P.; Vigo, D. 2001. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics (SIAM). 367 p.

Turskis, Z.; Zavadskas, E. K. 2010. A new fuzzy additive ratio assessment method (ARAS-F). Case study: the analysis of fuzzy multiple criteria in order to select the logistic centers location, *Transport* 25(4): 423–432. http://dx.doi.org/10.3846/transport.2010.52

Tušar, T.; Korošec, P.; Papa, G.; Filipič, B.; Šilc, J. 2007. A comparative study of stochastic optimization methods in electric motor design, *Applied Intelligence* 27(2): 101–111. http://dx.doi.org/10.1007/s10489-006-0022-2

Wang, X.; Tang, L. 2009. A population-based variable neighborhood search for the single machine total weighted tardiness problem, *Computers and Operations Research* 36(6): 2105–2110. http://dx.doi.org/10.1016/j.cor.2008.07.009

Wilcoxon, F. 1945. Individual comparisons by ranking methods, *Biometrics Bulletin* 1(6): 80–83. http://dx.doi.org/10.2307/3001968

Xing, L.-N.; Chen, Y.-W.; Wang, P.; Zhao, Q.-S.; Xiong, J. 2010. A knowledge-based ant colony optimization for flexible job shop scheduling problems, *Applied Soft Computing* 10(3): 888–896. http://dx.doi.org/10.1016/j.asoc.2009.10.006

Yi, W.; Kumar, A. 2007. Ant colony optimization for disaster relief operations, *Transportation Research Part E: Logistics and Transportation Review* 43 (6): 660–672. http://dx.doi.org/10.1016/j.tre.2006.05.004

Zeng, L.; Ong, H. L.; Ng, K. M. 2007. A generalized crossing local search method for solving vehicle routing problems, *Journal of the Operational Research Society* 58(4): 528–532. http://dx.doi.org/10.1057/palgrave.jors.2602171