The International Journal of
TRANSPORT &*LOGISTICS*
Medzinárodný časopis
DOPRAVA A LOGISTIKA

ISSN 1451-107X

VARIABLE NEIGHBORHOOD SEARCH ALGORITHM FOR GRAVEL UNLOADING DEVICES ALLOCATION PROBLEM

***Nenad Bjelić*[1], *Milorad Vidović*[2], *Branislava Ratković*[3]**

[1] *University of Belgrade, Serbia, tel: +381 11 3091273, e-mail: n.bjelic@sf.bg.ac.rs*

[2] *University of Belgrade, Serbia, tel: +381 11 3091202, e-mail: mvidovic@sf.bg.ac.rs*

[3] *University of Belgrade, Serbia, tel: +381 11 3091273, e-mail: b.ratkovic@sf.bg.ac.rs*

Abstract: *Gravel distribution by inland waterway transport is characterized with very intensive material flows and very low profit margins. Therefore, gravel distribution companies are in constant efforts on improving system efficiency by increasing productivity of existing facilities. Barges' unloading activity is executed at the end of inland waterway transport process and has a large influence on productivity of whole system, since minimization of time that barges spend waiting for services maximize time they transport gravel, i.e. their productive time. In this paper, we transfer this optimization problem in a well known class of multiple traveling repairman problem with time windows. Due to time consuming solution process in case of real size problem instances, beside optimal mathematical formulation of the problem we present solution approach based on Variable Neighborhood Search metaheuristic algorithm.*

Key words: *Gravel unloading, Optimization, Handling devices allocation, VNS*

1 INTRODUCTION

Gravel distribution by inland waterway transportation includes three main phases: loading of gravel by a suction dredger into barges, transport of gravel to the ports or unloading locations, and unloading of gravel by a handling facility that usually consists of pontoon mounted crane and belt conveyor. Because of high costs, a number of handling facilities is usually relatively small, and requires successive relocation of handling equipment between different unloading locations. Accordingly, providing efficient and cost effective service of loaded river barges needs appropriate allocation plan for handling equipment, which means defining sequence of unloading locations that should be served by each handling device.

The problem may be introduced in following way. For a given collection of barges unloading tasks find a set of assignments to minimize the sum of the service times including waiting for service and handling devices transfer times. The problem of this type may be

considered as dynamic handling devices allocation problem, where task occurrence times are distributed over a planning horizon, or as static problem, where all tasks are already present in the system at the beginning of a planning horizon. In this paper we studied generalization of the problem, i.e. dynamic handling device allocation problem (DHDAP) where all devices have different relevant characteristics.

The objective of this problem is minimization of barges service times so that they can spend as more as possible time in transporting goods, i.e. by making profit to an owner.
Remaining of the paper is organized as follows. In the section 2 we give problem formulation and relevant literature review. In section 3 we present variable neighborhood search algorithm for solving DHDAP. Framework for generating test instances is presented in section 4, while brief comments on obtained results and concluding remarks are presented in section 5.

2 PROBLEM FORMULATION

In case of homogeneous set of m handling devices $V' = \{1,2,...,m\}$, DHDAP is formulated on a complete, directed and asymmetric graph $G = (N, E')$, where $N = \{0,1,2,...,n+1\}$ is a set of nodes in which nodes 0 and n+1 correspond to the depot and $P = N \setminus \{0, n+1\}$ to the set of unloading tasks. The set $E' = \{(i, j): \forall i, j \in N\}$ is set of edges. Weight t'_{ij} is associated to each edge $(i, j): \forall i, j \in N$ representing traveling time of a device over the edge. To each node $i \in P$ there is a time window associated with it. However, in the case of DHDAP time windows impose only the earliest time of beginning of unloading service, defined by barge occurrences. Therefore, there is only left hand side of time window of task i, represented with parameter e_i, while the right side (l_i) is supposed to be infinite. Additionally, unloading service time, s'_i, is also associated to each node $i \in P$. The objective of homogeneous DHDAP is to minimize the sum of time all nodes wait until the end of service by identifying set of m device routes such that all nodes from P are visited exactly once by exactly one route while respecting time window constraints and service times. Device routes start at node 0 and end at node n+1. Parameters s and e for set of nodes $\{0, n+1\}$ are equal to zero.

However, since unloading devices differ in both traveling and unloading speed, previously formulated problem must be generalized by considering heterogeneous set of m devices $V = \{1,2,...,m\}$. Therefore, heterogeneous DHDAP is formulated on a graph G where set of edges E' is replaced with set $E' = \{(v, i, j): \forall v \in V, \forall i, j \in N\}$ meaning that between each pairs of nodes $(i, j) \forall i, j \in N$ there are m different edges exclusively dedicated to each member of V. Accordingly, weight t^v_{ij}, representing traveling time of a device v between nodes i and j, is associated to each edge in E, as well as unloading service time of device v, s^v_i, is associated to each node in N. The objective of the heterogeneous DHDAP is the same as in the homogeneous case.

Due to cumulative nature of objective function, DHDAP is very similar to the set of problems known in literature by names traveling repairmen problem (TRP), delivery man problem, school bus routing problem, minimum latency problem and cumulative capacitated vehicle routing problem.

As for the literature on DHDAP, there are only a few papers regarding gravel unloading devices control. In the [1] authors formulated the Handling Devices Allocation Problem – HDAP. They, considered static case of HDAP (SHDAP) and presented two approaches for its mathematical formulation. First one is based on three-dimensional assignment problem, while

the second one is based on similarities of SHDAP and the Static Berth Allocation Problem.. Additionally, authors presented a three step heuristic algorithm to solving SHDAP (CLASORD – CLustering ASsignment ORDering). [2] and [3] formulated dynamic version of the problem (DHDAP) and gave two mathematical formulation of the problem.

When similar problems, such as class of the TRP problems, are respected, situation about previous researches is not much better. Namely, the literature on mTRPTW, which is the closest problem to the heterogeneous DHDAP, is very limited, as well. To the best of our knowledge the only paper regarding mTRPTW is the one from [4] in which the author solved the problem of operational control of automated guided vehicles (AGV) fleet for different conditions of internal transportation system, such as off-line and on-line control, dwell point strategies, demand intensities, etc. In case of off-line control author proposed mTRPTW mixed integer linear programming (MILP) model for homogeneous fleet of AGVs. Since mTRPTW, as generalization of TRPTW, is NP-hard proposed model is used only for small problem instances. For medium and large instances author proposed insertion based algorithm.

Literature regarding TRPTW is also very limited and consists of two papers. In the first one, [5], author presented polynomial algorithms or NP-completeness for some special cases of TSPTW and TRPTW. Author showed that in case of TRPTW in which only tasks' release times are imposed, which is the case of the problem considered in this paper, problem is strongly NP-complete even if number of nodes is bounded to one. In the second paper, [6], authors give arc flow and sequential assignment based MILP formulations of the TRPTW. In a case of the second formulation in which all time windows are open, except the one in a depot, authors performed polyhedral study. For solving TRPTW authors presented both, exact and heuristic algorithms.

From previously said it is obvious that HDAP belongs to a class of complex optimization problems whose effective implementation in solving real system problems implies development of efficient (meta)heuristic algorithms. Therefore, in this paper we present relatively new solution procedure based on systematic changes of solution's neighborhood structures and compare it's performances to two other solution procedures.

3 VARIABLE NEIGHBORHOOD SEARCH METAHEURISTICS

VNS is relatively new [7,8,9] metaheuristic framework for developing heuristics algorithms that has intensively been used in solving variety of combinatorial optimization problems. It is based on straightforward facts that local optimum of one neighborhood structure does not have to be local optimum of some other neighborhood structure, that global optimum is local optimum of all neighborhood structures, and that for many problems local optima of one or several neighborhood structures are close to each other. Therefore, due to the expectation of finding improved solution in a neighborhood structure of the current solution x, VNS is based on systematical exploration of $k_{\max}$ neighborhood structures of the current solution ($N_k(x), k = 1,2,\ldots,k_{\max}$).

Neighborhood structures are changed sequentially until better solution is found. Afterwards, N_1 structure is explored with respect to the new best solution. However, exploration of k^{th} structure is realized in one of three different ways: deterministic, stochastic and both deterministic and stochastic. In the deterministic case, exploration of $N_k(x)$ is executed either until the local optimum of the structure is found or only until the first better solution is reached. The former case is called best improvement strategy, while the latter one is called first improvement strategy. This type of neighborhood structure search algorithms is

called variable neighborhood descent (VND) algorithm and its first improvement variant, implemented in this research, is shown in form of a pseudo code as algorithm 1.

Stochastic exploration of $N_k(x)$ implies selection of random point (x') within N_k and comparison with current best solution (x). If x' is not better than x, neighborhood structure is changed ($k \leftarrow k+1$) and new random point, now within $N_{k+1}(x)$ is selected. This type of VNS algorithm is called reduced VNS (RVNS) because, oppositely from VND, it does not perform deep exploration of neighborhood structures around x'. This drawback is eliminated by including VND as a part of the algorithm. Combined algorithm is known as a basic VNS and represents mixture of deterministic and stochastic exploration of neighborhood structures. Framework of basic VNS used in this paper is shown in form of pseudo code as algorithm 2.

Algorithm 1: First improvement Variable Neighborhood Descent algorithm

Initialization Select set of neighborhood structures, N_l'' $(l = 1,2,\ldots,l_{\max})$, to be used in search; find an initial solution x'.
Set $l \leftarrow 1$
Repeat following steps until $l \le l_{\max}$
(a)*Exploration of neighborhood*. If $N_l'' \neq \phi$, select next neighbor, x'', from N_l'' $(x'' \in N_l''(x'))$; else, set $l \leftarrow l+1$
(b)*Move or not*. If x'' is better then x', set $x' \leftarrow x''$, $l \leftarrow 1$; else set $N_l'' \leftarrow N_l'' \setminus \{x''\}$

VNS performs extensive exploration of solution space regions by executing VND algorithm, while trap of falling into local optima is avoided by implementing RVNS algorithm, i.e. by random selection of regions within neighborhood structure of currently the best solution. Procedure of random selection of regions is called shaking, or perturbation. It is important to emphasize that neighborhood structures used in VND algorithm and shaking procedure does not have to be identical and that final solution is optimum with respect to neighborhood structures from both VND and shaking procedures.

Another important aspect of implementing VNS is neighborhood structure relation. Namely, although from VNS pseudo code it can be concluded that neighborhood structures are independent, sequences of nested neighborhood structures are frequently implemented. Such structures imply that each structure in sequence is subset of the following structure, i.e. $N_1 \subset N_2 \subset \cdots \subset N_{k_{\max}}$.

Algorithm 2: Basic Variable Neighborhood Search algorithm

Initialization Select set of neighborhood structures, N_k $(k = 1,2,\ldots,k_{\max})$, to be used in search; execute RVNS for obtaining an initial solution x; choose stopping condition
Set $k \leftarrow 1$
Repeat following steps until $k \le k_{\max}$
(a)*Shaking*. Select a solution x' from neighborhood structure of x, $x' \in N_k(x)$
(b)*Local search.* Apply some local search (VND) with x' as initial solution. Obtained local optimum is marked as x''.
(c)*Move or not*. If x'' is better then x, set $x \leftarrow x''$, $k \leftarrow 1$; else set $k \leftarrow k+1$

Like in many other problems that imply use of multiple task executers, DHDAP solutions are represented as ordered sets of task indices - H^v served at m devices where stands $\sum_{v\in V}\left|H^v\right| = n$. $\left|H^v\right|$ denotes cardinality of set H^v, i.e. number of tasks served by device v. Objective function of a solution $f(x)$ is calculated according to the expression (1).

$$f(x) = \sum_{v=1}^{m}\sum_{i=1}^{\left|H^v\right|} D_{h_i^v} - r_{h_i^v} \tag{1}$$

Where notation is the same as in section 2 and h_i^v refers to the index of the task that is on the i^{th} position in set H^v. For example, if $H^3 = \{3,5,2,6\}$ then $h_1^3 = 3$, $h_2^3 = 5$, $h_3^3 = 2$ etc. Task finish times, $D_{h_i^v}$, are calculated according to (2)

$$D_{h_i^v} = \begin{cases} \max\left(A^v + \dfrac{d_{0\,h_i^v}}{v_v}, r_{h_i^v}\right) + s_{h_i^v}^v, \; i = 1 \\ \max\left(D_{h_{i-1}^v} + \dfrac{d_{h_{i-1}^v\,h_i^v}}{v_v}, r_{h_i^v}\right) + s_{h_i^v}^v, \; i > 1 \end{cases} \tag{2}$$

Due to their wide use, a lot of solution transformations forming neighborhood structures can be found in a literature. All of neighborhood structures we use in our VNS are already known structures whose implementation is improved by adjustment to mTRPTW. VND algorithm we use in this paper explores three independent neighborhood structures generated by heuristics found in[10]:

Insertion heuristics includes execution of Or-opt moves on each device's set of tasks with aim to improve solution by reordering current sequence of execution. Reordering implies removal of one, two and three adjacent tasks from existing execution sequence and their insertion to all possible positions in remaining task sequence, excluding their original position. All possible moves of this type, performed for each device separately, form neighborhood structure N_1''. Figure 1a presents example in which tasks *j* and *k* served after task *i* prior to insertion move are inserted between tasks *j* and *m* afterwards insertion task.

Swap heuristics (Figure 1b), like in case of N_1'', keeps the same structure of the solution regarding number of tasks served on each device. Swap move tries to find improved solution by exchanging a task served on its device with tasks served on all other devices. Execution of this move for each task on all devices generates neighborhood structure N_2''. An example of swap move is presented on figure 3b where it can be seen that after swap move tasks *k* and *l* exchanged positions in sequences on which they ware served before swap move.

Relocation heuristics (Fig 1c) forms neighborhood structure N_3'' by changing existing structure of currently the best solution. Namely, relocation move tries to improve solution by removing a task form a device where it is served and inserting it at all possible locations on other devices. Structure N_3'' is formed after relocation move is executed for all tasks in the best solution. An example of relocation move is presented on figure 3c where task *l*, served between tasks *j* and *n*, is moved to be served by another device between tasks *k* and *m*.

For shaking procedure we use two sets of nested neighborhoods:

Random swap heuristics chooses next region for exploration by random selection of two tasks for interchange. This procedure is expansion of swap heuristics from VND since set of tasks for interchange with selected task is not limited to tasks served by other devices. Nested neighborhood structures N_k' ($k = 1,2,\ldots,k_{\max}^1$) are formed by successive implementation of random swap algorithm.

Worst relocation heuristics selects next regions for local search in two steps. In the first step, a task with the largest waiting time till the end of service (worst task) is removed from existing solution. Afterwards, from set of all possible insertions, excluding insertion to the previous location, removed task is inserted into position with the lowest objective function value. Like in case of random swap heuristics nested neighborhood structures N'_k ($k = k^1_{\max}+1, k^1_{\max}+2, \ldots, k^1_{\max}+k^2_{\max}$) are formed by successive execution of worst relocation heuristics.

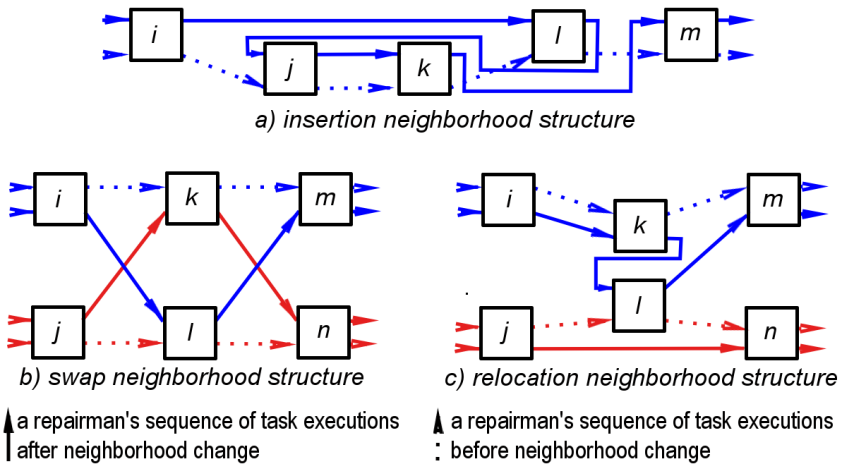

Fig. 1 *Neighborhood structures used in VND algorithm*

Numbers of nested structures, $k^1_{\max}$ and $k^2_{\max}$, as in case of MSVND algorithm, are determined from results of pilot studies resulting in best performances for values of parameters equal to eight.

4 COMPUTATIONAL EXPERIMENTS

Set of 45 randomly generated instances is formed on example of Belgrade service area (figure 2) consisting of 20 gravel unloading locations. We considered cases with equal chance of task arrivals at each location and with uniform distribution of task inter arrival times over planning horizon. Each task implies need for unloading 1000 tons of gravel. Devices used for gravel unloading belong to one of three different classes whose relevant characteristics are presented in table 1. Following series of device classes {I, II, III, II, III, II, III, I, II, III} corresponds to classes of devices making fleets of 2, 5 and 10 devices used for executing unloading tasks. It is supposed that all devices are available for service at the beginning of planning horizon.

Tab.1 *Characteristics of unloading device classes*

Class	Transfer speed [km/h]	Unloadingcapacity [t/h]
I	6	100
II	5	150
III	4	200

Beside VNS algorithm, efficiency of Insertion heuristics [4] is tested as the only heuristic algorithm used so far for solving mTRPTW. Additionally, we tried to obtain optimal solutions

of small size problem instances (10 tasks) by using branch and bound (B&B) algorithm implemented in CPLEX 12.2 with limits of 512MB of RAM for tree structure and one hour for running time. However, due to high complexity of the problem only one optimal solution, bolded in table 2, is achieved.

Fig.2 *Spatial distribution of gravel unloading location in the case of Belgrade, Serbia*

All test runs are executed on Windows XP OS powerd by AMD Phenom II 2.61 GHz processor with 1GB of RAM while all coddings are done in Python 2.5.

Tab.2 *Results of instances containing 10 unloading tasks*

			B&B		VNS		Insertion	
m	**inst**	**objective**	**Δobj [%]**	**T [s]**	**Δobj [%]**	**T [s]**	**Δobj [%]**	**T [s]**
2	1	195.3823	0.0%	3008.44	0.0%	0.80	0.000%	<0.01
	2	189.0527	0.0%	3201.25	0.0%	0.78	1.397%	<0.01
	3	202.1763	0.0%	3058.92	0.0%	0.78	0.542%	<0.01
	4	196.2127	0.0%	3375.86	0.0%	0.81	0.356%	<0.01
	5	181.267	0.0%	3689.45	0.0%	0.87	7.457%	<0.01
	Average		0.0%	3266.78	0.0%	0.81	1.95%	<0.01
5	1	64.17417	0.0%	1585.16	0.0%	0.92	6.042%	0.02
	2	62.10733	0.0%	1487.70	0.0%	0.87	10.896%	<0.01
	3	65.61867	3.08%	2011.20	0.0%	0.86	5.547%	<0.01
	4	66.52917	0.0%	1821.41	0.79%	0.88	13.649%	<0.01
	5	59.65999	0.0%	1280.09	0.0%	1.00	5.008%	<0.01
	Average		0.62%	1637.11	0.16%	0.90	8.23%	0.004
10	1	54.62417	0.388%	1908.63	0.0%	1.15	1.624%	<0.01
	2	54.43017	1.492%	158.20	0.0%	1.04	1.174%	<0.01
	3	53.30733	0.0%	6075.91	1.61%	1.02	6.893%	<0.01
	4	54.48233	0.0%	3936.48	0.0%	1.08	1.285%	0.02
	5	51.7057	0.0%	3026.53	1.8%	1.21	1.801%	0.02
	Average		0.38%	3021.15	0.68%	1.10	2.56%	0.01

In cases of B&B and Insertion algorithms instances are solved once, while in the case of the VNS, due to stochastic nature of algorithm, solution procedure is repeated five times. Summarized results for small size problem instances are presented in table 2, while results for larger instances are given in tables 3 and 4.

In case of larger problem instances, number of tasks considered in an instance is given in column "n" of tables 3 and 4. Average time of an algorithm run is presented in column "T" of table 2, as well as in table 4, for larger problem instances. Table 2's column "objective" contain values of the best objective values of instances (f_{best}), achieved over all instance runs for considered fleet size. Relative gap of an algorithm's best result (f) from f_{best} is given in column "Δobj", for each algorithm. "Δobj" is calculated by expression (3).

$$\Delta\text{obj} = \frac{f - f_{best}}{f_{best}} \cdot 100\% \tag{3}$$

Table 3 contains only data about best solution found by appropriate procedure. However, because VNS outperformed Insertion heuristics for every problem instance, instead using "objective" column, like in case of table 2, part of table 3 regarding VNS's results contains values of the best solution's objective function. Part of table 3's results, regarding results of Insertion algorithm, contains values of relative gap to the solution of VNS, i.e. it contains information like in "Δobj" column of table 2. Table 4 contains information about average time of algorithm runs for larger problem instances, like in "T" column of table 2.

Tab.3 *Values and gaps[%] of best achieved solutions*

		VNS			**Insertion**		
n	**inst**	**2**	**5**	**10**	**2**	**5**	**10**
25	1	928.77	165.99	132.68	4.43%	14.92%	0.84%
	2	967.03	177.73	137.57	2.46%	19.96%	2.30%
	3	1001.60	181.80	136.70	7.82%	30.86%	2.43%
	4	996.00	183.53	136.10	5.79%	16.21%	1.85%
	5	960.61	182.30	135.72	5.06%	20.12%	3.10%
				Average[%]	5.11%	20.41%	2.11%
50	1	3660.22	389.89	276.02	4.00%	44.68%	1.82%
	2	3461.11	343.03	271.03	1.57%	28.51%	2.57%
	3	3690.70	396.57	276.23	3.66%	41.99%	2.59%
	4	3727.83	424.44	273.24	4.01%	35.87%	2.60%
	5	3527.05	384.18	273.04	3.22%	33.16%	3.04%
				Average[%]	3.29%	36.84%	2.53%

Tab.4 *Average running times for large size instances [s]*

		VNS			**Insertion**		
n	**inst**	**2**	**5**	**10**	**2**	**5**	**10**
25	1	15.01	15.28	15.52	0.01	0.02	0.03
	2	19.20	13.70	9.49	0.02	0.02	0.02
	3	19.97	18.72	10.75	0.01	0.02	0.02
	4	17.98	17.46	11.98	0.02	0.01	0.02
	5	16.47	17.90	15.72	0.01	0.02	0.02
Average		17.73	16.61	12.69	0.015	0.02	0.02
50	1	348.36	171.84	226.65	0.08	0.09	0.11
	2	313.21	250.41	176.39	0.08	0.08	0.11

3	385.83	192.77	171.64	0.08	0.08	0.11
4	176.27	195.59	205.05	0.08	0.09	0.09
5	344.38	267.43	183.60	0.08	0.09	0.11
Average	313.61	215.61	192.67	0.08	0.09	0.1

6. CONCLUDING REMARKS

From content of table 2 it can be noticed the influence parameter m, i.e. number of available unloading devices, has on complexity of the DHDAP. Namely, as m increases complexity of the problem increases as well, so that B&B algorithm was not able neither once to solve problems with ten tasks to optimality for the cases when m was larger then two. However, beside larger solution space, increase of parameter m, i.e. increase of the number of devices in system, causes potential existence of multiple optimum solutions. Especially in cases when capacity of available devices exceeds required capacity and if there are several devices with the same relevant performances, meaning that mutual changes of devices leads to the same objective functions. Therefore, due to larger number of optimal solutions in solution space there is slight larger chance of finding it. According to that, improvement of Insertion's results for the case of m=10 compared to other m values is not surprising.

As it was expected, in cases of larger problem instances, VNS outperformed the Insertion heuristics in all problem instances. This performance indicates necessity of further research efforts in order to improve quality of obtained solutions, as well as to reduce running time of VNS.

Since motivation for researching this problem originated from need of controlling system for unloading gravel from barges, intentions for future research are related to implementation of algorithms in such a system. That implies future research should be focused on additional increase of time efficiency of VNS algorithm, which might be achieved by consideration of additional neighborhood structures. Next direction for future research is related with testing efficiency of algorithms belonging to other classes of metaheuristic algorithms, for example genetic algorithms, ant colony optimization, bee colony optimization, particle swarm optimization etc. Finally, third direction for future researches related to DHDAP makes it closer to decisions made by dispatcher in real systems, because it includes inventory management within the problem formulation. Namely, in order to eliminate inventory shortages on unloading locations, dispatcher controlling a fleet of unloading devices makes device to task allocation decisions not only by respecting unloading devices and barges related information, but also by respecting information related to inventory levels and consumption rates of unloading locations. By this kind of decision making, quality of service is increased, but on the other side obtaining of efficient solutions becomes more complicated.

ACKNOWLEDGMENT

This work was partially supported by Ministry of Science and Technological Development Republic of Serbia, through the project TR 36006, for the period 2011-2014.

References

[1] Vidovic M., Vukadinovic K., Allocation planning of handling devices for barges unloading, Proceedings of the 11th Meeting of the EURO Working Group on Transportation, Bari, (2006):740–747
[2] Bjelic N., Operational Control Models of Material Handling Systems with Distributed Resources, (2009) MSc Thesis (in Serbian), University of Belgrade, Serbia
[3] Bjelic N., Vidovic M., Miljus M., Two mathemtical modells of the handling devices allocation problem, Proceedings of the XXXVII Serbian Symposium on Operations Research SYM-OP-IS 2010, Tara 21st to 24th September, (2010):349-352
[4] van der Meer, R. Operational Control of Internal Transport (2000), PhD Dissertation, University of Roterdam
[5] Tsitsikils J Special cases of traveling salesman and repairman problem with time windows, Networks (1992) 22:263-282
[6] Heilporn G, Cordeau J, Laporte G, The Delivery Man Problem with time windows. Discrete Optimization (2010) 7: 269-282
[7] Mladenovic N, Hansen P Variable neighborhood search. Computers and Operations Research (1997) 24 (11):1097-1100
[8] Hansen P, Mladenovic N Variable neighborhood search: principles and applications. European Journal of Operational Research (2001),130:449-467
[9] Hansen P, Mladenovic N Variable neighborhood search. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics (2003). Kluwer Academic Publisher, pp 145-184
[10]Hansen P, Oguz C, Mladenovic N Variable neighborhood search for minimum cost berth allocation, European Journal of Operational Research (2008) 191:636-649