International Journal for Traffic and Transport Engineering, 2021, 11(2): 280 - 293
DOI: http://dx.doi.org/10.7708/ijtte2021.11(2).08 *UDC: 005.5:339.372*

QMIP AND HEURISTIC APPROACH IN SOLVING AISLE CONGESTION PROBLEM BY REALLOCATING GOODS WITHIN AN ORDER PICKING ZONE

Nenad Bjelić[1], Milorad Vidović[2], Branislava Ratković[3]
[1,2,3] *University of Belgrade, Faculty of Transport and Traffic Engineering, Logistics Department, Vojvode Stepe 305, 11000 Belgrade, Serbia*

Received 30 November 2020; accepted 11 February 2021

Abstract: Because order picking is the most demanding, the most labor intensive and, accordingly, the costliest activity in a warehouse it has been potential for the optimization from different points of view and at different decision-making levels. In this research our goal was to examine a possible improvement of the order picking process by reducing the potential of the aisle congestion which is quite frequent in order picking systems with lots of pickers passing through same isles simultaneously. For that purpose, we give a quadratic mixed integer programming solution approach, as well as a Variable Neighborhood Search based heuristic algorithm solution approach. Beside that we employ a reduction-based strategy for reducing running times of the heuristic algorithms. Eventually, we tested the efficiency of developed models on an imaginary order picking system.

Keywords: warehouse, order picking, quadratic mixed integer programming, VNS.

1. Introduction

According to de Koster *et al.* (2007) "order-picking (OP) is the process of retrieving products from storage (or buffer) areas in response to a specific customer request". Practically every aspect of the OP can be realized in numerous ways (e.g. man-to-goods vs. goods-to-man, sequential vs. batched OP, random vs. zoned OP, etc.) so that every OP system is pretty much unique. For the deeper insight into the details of the OP we refer interested reader to Djurdjević (2019), Gu *et al.* (2010), Dallari *et al.* (2009), Hompel and Schmidt (2007) or Wascher (2004). Nevertheless, due to its simplicity and robustness, one of the most widely implemented OP technology is a sequential men-to-goods OP strategy within an OP area completely separated from the reserve storage zone, or as the lowest level(s) of the storage zone racks. This OP technology implies that every customer demand is fulfilled by one order-picker which goes between products locations within an OP area and picks up a required quantity of products. Order picker starts from the starting location and finishes and the end location which may be spatially separated. At the end location shipment is prepared (checked, consolidated, labeled etc.) for the next step in the delivery process.

Because OP is the most demanding, the most labor intensive and accordingly the costliest activity in a warehouse, with the contribution

[1] Corresponding author: n.bjelic@sf.bg.ac.rs

of around 50% in the warehouse operating costs (Frazelle, 2001), the OP process has been potential for the optimization from different points of view and at different decision-making levels. However, due to the limited space and numerous papers dealing with order picking related problems, for detailed overview of considered problems, used parameters and implemented solution techniques we direct interested reader to van Gils *et al.* (2018).

In this research our goal is to examine a possible OP improvement by reducing the aisle congestion problem which is frequent in OP systems with lots of order-pickers that passes through same isles at the same time. In the following section we give more details about the considered problem. In the Section 3 we give mathematical formulation of the problem, as well as Variable Neighborhood Search based algorithms for solving it. In Section 4 we give settings and results of conducted numerical experiments. Eventually, in Section 5 we give overview of possible future directions of the problem.

2. Problem Description

In order to provide timely service, usually numerous order-pickers are engaged in the realization of the OP process. On the other side, warehouse space utilization implies that aisles are as narrow as it is imposed by technical features of the handling equipment. Consequentially, large number of pickers, that are moving within aisles at the same time, is causing congestion that eventually might significantly reduce overall OP performances. In this research, situations that are considered as congestion from a picker's perspective can be generally divided into two subcategories. The first one implies blockage of a picking location(s) due to an occupation by another picker with enough space in the aisle for passing that picker. This situation is showed on the Figure 1a where it can be seen that, due to the additional handling equipment used in the OP process, neighboring locations of the picking location may also be blocked. Depending on the location of the picking position and the size of the handling equipment, a number of blocked positions can be only one, in the case when location is at the beginning/end of an aisle and with the policy that handling equipment can be placed in the cross isle, or when the equipment size gives the possibility to be placed in such a way to block only one position. However, in the situation when equipment with the capacity of two pallets is used the number of blocked positions can go up to three.

The second category of considered congestion situation is given on the Figure 1b and the Figure 1c where it can be seen that beside blockage of picking location, and its neighboring positions, due to the simultaneous existence of pickers on sufficiently close distance on both sides of an aisle, all downstream picking positions in an isle are blocked.

One approach in avoiding beforementioned congestion situations is exploiting the dynamic nature of the OP process. This means that assignments of tasks to pickers is realized in such way that situations that lead to congestion are minimized by selecting appropriate OP tasks, routes, or moments in which pickers will start their routes. In the literature there are several studies that implemented this operational

level approaches in minimizing congestion. In that sense, for example Bataineh and Khasawneh (2016) use taboo search metaheuristic algorithm to find the routes of order pickers with the goal to minimize a number of congestion of occurrences. However, authors by congestion consider only Pick-Face Blocking situation which occur if two, or more, pickers as interested in picking from the same picking location (i.e. within the same vertical bay).

However, operational level approaches are not the only way in which the congestion problem could be minimized. In that sense, in this research we propose an approach which utilizes a tactical level measure of reallocation of storage keeping units (SKU) over the OP zone based on the historical data about pickers delaying at SKUs' locations. To the best of our knowledge this approach has not be used so far and together with the operational measures it could additionally reduce negative effects of the considered congestion problem. The main idea of the occasional reallocation of SKUs is to reduce the chance of congestion situations by optimizing locations of all SKUs. Namely, based on the congestion case from Figure 1a it is obvious that locating SKUs with high picker delay times next to each other will probably lead to frequent occurrences of situations which imply blocking of SKUs. Accordingly, SKUs should be allocated in such way that SKUs with long and short picker delay time are located alternately in an aisle. Moreover, since locating two SKUs with long picker delay times directly across one another in an aisle leads to frequent congestion situations form Figures 1b and 1c, it is obvious that allocation of SKUs must be realized by simultaneous consideration of all SKUs' picker delay times, i.e. not only on a side of the aisle in which an SKU is to be allocated, but also with the respect to the SKUs to be allocated on the opposite site of the aisle. Nevertheless, since SKU's that are allocated on the opposite side of an isle and two positions from the considered location will not cause aisle blockage, but rather congestion situation given on the Figure 1a, we define the congestion zone for each picking location as in Figure 2. Accordingly, each congestion zone consists of the zone center, i.e. considered picking location, and zone members, i.e. neighboring locations of the picking location. Situations for the case of picking locations at the beginning/end of an aisle and in the middle of an aisle location are given on the Figure 2a, and Figure 2b, respectively.

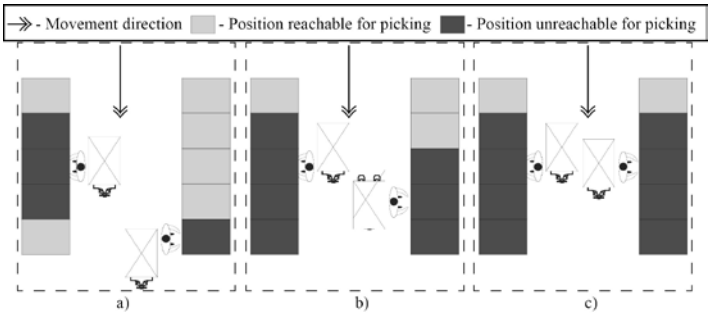

Fig.1.
Considered Congestion Situation

As a performance measure that is minimized in this research we use the sum of all congestion zone times whose number is equal to the number of picking locations. Congestion zone times are calculated as the sum of picker delay times for all members of the zone plus zone center. Common situation in warehouses is that building pillars are located within appropriate picking locations, meaning that no SKU can be allocated to such locations. Accordingly, such situation is handled by this allocation approach by straightforward pre-allocation of dummy SKUs with zero picker delay time to such locations.

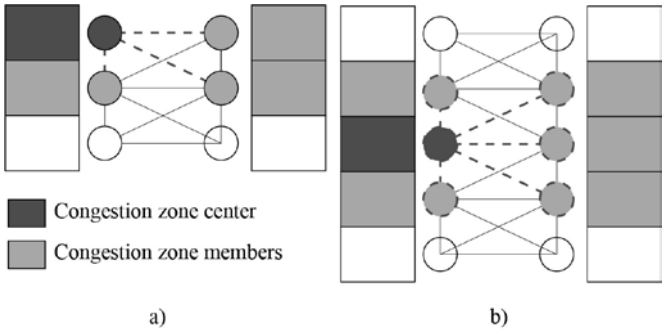

Fig.2.
Congestion Zone Examples

3. Solution Methods

The defined problem can be given in a graph form by representing picking locations as nodes of the graph which includes connection of neighborhood picking locations by arcs (Figure 2). It should be noted that in general sense by neighborhood picking locations we consider location which are one position up and down the moving direction on both sides of an aisle, as well as position directly across the considered congestion zone center (i.e. considered picking position). Accordingly, graph representation of the problem consists of P (P is the number of aisles in a warehouse) disconnected subgraphs where each subgraph represents picking locations in one warehouse aisle. Based on this graph representation of the problem we define two approaches for its solving.

3.1. Quadratic Mixed Integer Programming Model

Based on all previously said we define allocation model based on utilization of the classic formulation of the assignment problem. However, due to the nature of the performance measure we add additional set of constraints, and change the objective function which eventually changes its linear nature.

Sets:
L – set of all picking locations (l) within a warehouse
A - set of all SKUs, or articles (a), to be located in the L, $|L|=|A|$
l_z – set of congestion zone nodes for the node l (includes location l and all neighborhood locations)

Parameters

t_a – expected picking time delay of the SKU a in the considered planning horizon.

Decision variables:

$$x_{a\_l} = \begin{cases} 1 \text{ - SKU } a \text{ is allocated at the location } l \\ 0 \text{ - otherwise} \end{cases}$$

c_l - expected picker's delay time in the congestion zone with the zone center l

By respecting all previous notations, the appropriate Quadratic mixed integer programming allocation model can be formulated as:

$$\min \quad \sum_{a \in A} \sum_{l \in L} x_{a\_l} \cdot c_l \tag{1}$$

Subject to:

$$c_l = \sum_{l^n \in l_z} \sum_{a \in A} x_{a\_l^n} \cdot t_a \qquad \forall l \in L \tag{2}$$

$$\sum_{a \in A} x_{a\_l} = 1 \qquad \forall l \in L \tag{3}$$

$$\sum_{l \in L} x_{a\_l} = 1 \qquad \forall a \in A \tag{4}$$

$$x_{a\_l} = \{0,1\} \tag{5}$$

Objective function (1) minimizes the overall sum of pickers' delay times in congestion zones. Constraint (2) defines values of the variable c_l. Constraint sets (3) and (4) are usual assignment constraints related to allocation of SKUs to picking locations. Finally, constraints (5) define binary nature of the variable x_{a_l}.

3.2. Variable Neighborhood Search based Approach

Regardless of the integrability property of the assignment problem, changes made in order to adopt the assignment problem to the considered problem, especially quadratic nature of the objective function, made this approach very inefficient for solving the problem. As it will be showed in the next Section solving time of not very large problems is inadmissibly long. Consequentially, in order to reduce the solving time of the problem, as well as to gain solution of better quality we developed a Variable Neighborhood Search (VNS) metaheuristic-based approach. VNS is a metaheuristic framework for developing heuristics algorithms (Hansen and Mladenović, 2014; Hansen and Mladenović, 2001; Mladenović and Hansen, 1997) for solving more specific problems. VNS have been intensively used in solving a variety of combinatorial optimization problems. It is based on straightforward facts that (1) a local optimum of one neighborhood structure does not have to be a local optimum of another neighborhood structure; (2) that a global optimum is a local optimum of all of the neighborhood structures; and (3) that, for many problems, the local optima of one, or several neighborhood structures are close to each other. Therefore, VNS is based on the systematic exploration of neighborhood structures of the current solution by performing the Variable Neighborhood Descent (VND) algorithm. The result of the VND execution is a local optimum in regard to considered neighborhoods. When

the VND results with solution improvement procedure is restarted with the improved solution in the center of the search.

In the cases when VND does not result in solution improvement, chance of finding a global optimum is increased by making a random selection of the solution space regions within the neighborhood structures of the current best solution. The procedure of the random selection of the regions is called shaking, or perturbation. It is important to emphasize that the neighborhood structures used in the VND algorithm and in the shaking procedure can be, but it is not mandatory to be the same as in the case of the VND. The framework of the VNS used in this paper is based on the General VNS algorithm (given in Hansen and Mladenović, 2014), and it is presented on Figure 3.

Algorithm 1. Variable Neighborhood Search Algorithms		
Initialization: Set of neighborhood structures N' consists of the crossing structure; generate an initial solution x; stopping condition is defined by the parameter *NoUnSrch*.		
While $u <= NoUnSrch$:		
Shaking:	non-nested variant	nested variant
	Generate a solution x' by executing crossing move n successive times with x as the incumbent solution.	Generate a new solution x' by executing crossing move on the existing x'.
Local search: Apply the VND algorithm with x' as initial solution. The obtained local optimum is marked as x''.		
Move or not: If x'' is better than x, set $x \leftarrow x''$ and $u \leftarrow 1$; otherwise, set $u \leftarrow u + 1$		

Fig.3.
Implemented Variant of the General VNS

3.2.1. Solution Representation and an Initial Solution

Solution of the problem in the VNS is given as the allocation of products to storage locations. Accordingly, the objective function of the problem is calculated by summing all zone picking delay times, where each zone delay time is obtained by adding expected time delays of the SKU allocated to the considered zone.

Initial solution of the problem is composed of two sets of SKUs. The first one relates to the SKUs that are already present in the system and will be present in the system in the following planning period, while the second one relates to SKUs that were not in the system in the previous planning period but will be in the following period. Accordingly, in the initial solution the members of the first set are allocated to their existing locations, while members of the second set are allocated to the remaining locations in a random manner.

3.2.2. Shaking Procedure

For the shaking phase of the VNS we used only one neighborhood structure, known in the literature as the crossing move. This move implies a random selection of two racks for SKUs exchange, where by a rack we consider all locations along one side of an aisle. Every exchange of SKUs is executed in such way that a random crossing location is

defined for both racks after which all SKUs from that point (including crossing location) to the beginning of the rack are exchanged between two included racks. In the case when number of locations within selected racks is not equal crossing location is bounded by the size of the shorter rack. Examples of the crossing moves for the cases of the same and different rack sizes are given on Figure 4a and 4b, respectively.

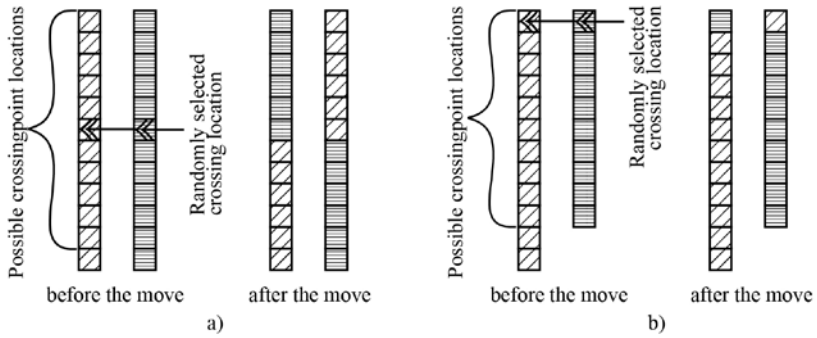

Fig.4.
Crossing Move Implemented in the Shaking Procedure

Because of its importance in overcoming the problem of falling in the local optima finding an efficient shaking is very important for obtaining an efficient VNS algorithm. Therefore, besides "regular" shaking we considered nested shaking concept. Difference between nested and regular concepts is in the solution they use for generation of the new solution. In the regular shaking procedure new solutions are always generated with the incumbent solution as the starting solution, i.e. all changes are executed on the incumbent solution. On the other hand, nested concept implies that a new solution is obtained by changing a solution obtained in the previous shaking procedure. Of course, in the first shaking procedure, as well as in all shaking procedures that follow the improvement of the incumbent solution in the local search phase of the VNS, nested procedure also considers the incumbent solution as a starting point for building new solution. Accordingly, two VNS algorithms are later tested and VNS algorithm with regular (non-nested) shaking is referred as nnVNS, while VNS algorithm with the nested concept is referred as the nVNS.

3.2.3. Stopping Criteria

Shaking procedure is usually closely related with the stopping criteria of the VNS algorithm. Namely, if there is no time limit imposed to the algorithm, the algorithm stops after the sequence of local search algorithms, realized on shaking solutions, unsuccessful in finding an improvement of the incumbent solution. The length of the sequence is defined by the parameter *NoUnSrch*. The value of the parameter *NoUnSrch* is the subject of optimization since higher values should result in better solutions, but usually requires more time. In that sense, in the following section we tested the influence of the *NoUnSrch* on the nnVNS and nVNS performances.

3.2.4. Variable Neighborhood Descent Local Search Algorithm

According to Hansen and Mladenović (2014) in the basic case of the VNS after the shaking move some kind of local search heuristics has to be applied for the case of a deeper search of the solution space around the shaking output solution. In the case when Variable Neighborhood Descent (VND) algorithm is used as the local search algorithm the general case of the VNS is generated, which is the variant of the VNS we implemented. The VND solution we implemented in solving the considered problem is given on Figure 5.

Generally speaking, we used two moves for generating neighborhood structures, but since one or two storage locations from each included rack are moved, formally we used four neighborhood structures. Nevertheless, in the rest of this section we explain moves with two engaged storage locations, while in Figure 5 we show all four moves. Of course, moves with one storage location are intuitive.

The first move that we used is well known Or-opt *n* move, where *n* denotes the number of storage locations included in the move. Or-opt move is well known from numerous combinatorial optimization problems among which it is the most widely used for the optimization of vehicle routing class of problems. The main purpose of the Or-opt move is in the optimization of the service order of tasks on a resource. In the case of the considered problem the role of the Or-opt move is the optimization of the allocation of SKUs to storage locations within a rack. Namely, Or-opt 2 move implies that for a selected rack two neighboring SKUs are taken from the current allocation and inserted on all possible locations in the remaining allocation structure (Figure 6a). Number of new solutions in the Or-opt *n* neighborhood structures is defined as $(N\text{-}n+1)(N\text{-}n)$ where *N* denotes the number of SKUs allocated in the considered rack.

The second move used in the VND is also well-known move from numerous combinatorial optimization problems where it is referenced either as the Exchange move, or as the Relocate move. From now on we refer it as the Exchange move. The purpose of the Exchange move is to optimize the solution of the problem related to the distribution of the tasks between resources. Analogously to the implementation of the Or-opt move, in the implementation of the exchange move resources are storage location of two considered racks while tasks are SKUs allocated in those racks. The Exchange 2 move implies that new solution is generated in such way that two neighboring SKUs from one rack (r_1) are selected and exchanged with two neighboring location from the second included rack (r_2), as in Figure 6b. Exchange neighborhood structure is generated by executing the Exchange move for all combinations of racks and all possible exchanges between SKUs in two included racks. If N_{r_1} and N_{r_2} denote numbers of SKUs in racks r_1 and r_2, respectively, and $|R|$ denotes the number of racks in a warehouse, then the size of the Exchange structure with *n* SKUs is $(|R|\cdot(|R|-1)/2)\cdot(N_{r_1}-n+1)\cdot(N_{r_2}-n+1)$.

Algorithm 2. First Improvement VND Algorithm

Initialization: Set of neighborhood structures N'' consists of Or-opt 2, Or-opt 1, Exchange 2 and Exchange 1structures; x' from shaking procedure is used as an initial solution, x_{loc_opt}; *OrCh* set is initialized with the set of racks, *R*; *ExCh* set is initialized as $ExCh \leftarrow \{(r_1, r_2) | r_1 \in R, r_1 \in R, r_1 \neq r_2\}$.

Beginning:
l←*2, impr*←False;
While 1<=*l*<=*2:*
  While *OrCh*:
    By random select the next rack *r* from *OrCh*; $OrCh \leftarrow OrCh \setminus \{r\}$
    Select next neighbor *x''* from Or-opt *l* neighborhood with rack *r*.
    If *x''* is better then x_{loc_opt}, set $x_{loc\_opt} \leftarrow x''$, *l*←2; *impr*←True, $OrCh \leftarrow OrCh \cup \{r\}$ and go to *the Beginning;*
    If *impr*←False set *l*←*1;*

l←*2, impr*←False;
While 1<=*l*<=*2:*
  While Ex*Ch*:
    By random select the next pair of racks, (r_1, r_2) from *ExrCh*; $ExCh \leftarrow ExCh \setminus \{(r_1, r_2)\}$
    Select next neighbor *x''* from Exchange *l* neighborhood for racks r_1 and r_2.
    If *x''* is better then x_{loc_opt}, set $x_{loc\_opt} \leftarrow x''$, *l*←2, *impr*←True, $OrCh \leftarrow OrCh \setminus \{r_1, r_2\}$; $ExCh \leftarrow ExCh \cup \{\{r_1 \times R \setminus \{r_1\}\}, \{r_2 \times R \setminus \{r_2\}\}\}$; go to *the Beginning;*
  If *impr*←False set *l*←*1;*

Fig.5.
Implemented Variant of the Variable Neighborhood Descent Algorithm

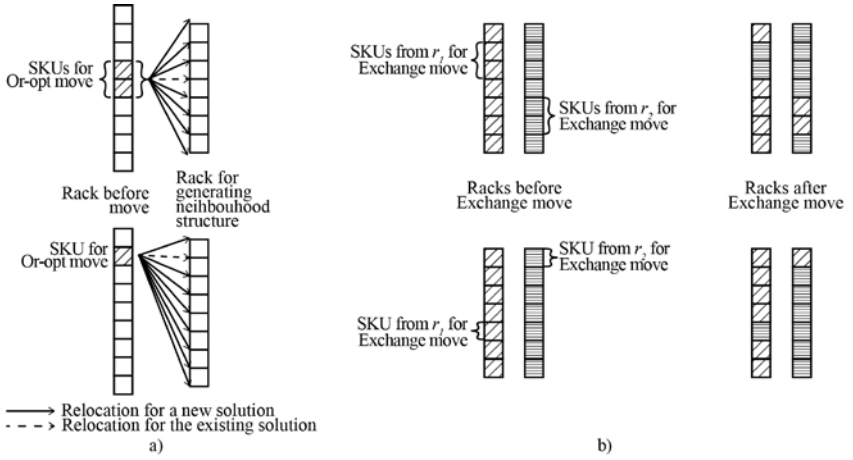

Fig.6.
Or-opt n and Exchange n Moves used in the VND

The order of implemented neighborhood structures in the VND is a very important aspect of the algorithm's implementation and in the implemented VND we obeyed the rule that neighborhood structures are used according to their sizes. Therefore, the order of neighborhood structures is Or-opt 2, Or-opt 1, Exchange 2 and finally Exchange 1. However, it should be mentioned that before exploration of neighborhood structures we firstly select rack(s) on which structures will be implemented. More precisely, in the case of Exchange *n* structures we firstly select a pair of racks on which sequence of Exchange 2 and Exchange 1 moves will be executed. In other words, we do not execute Exchange 2 for all pairs of racks and then start with the Exchange 1 move, but we execute Exchange *n* moves on a selected pair of racks and then move to the next pair of racks.

Previously mentioned order of combining neighborhood structures and selected racks is of a vital role for implementing a reduction strategy for the VND. Namely, by a general rule each time VND finds an improved solution it is restarted (it begins after the initialization step) which means that it explores neighborhood structures all over again, even for the structures that it already checked and which had no changes since the improvement of the solution. For example, VND would explore the structure of a rack that is the same as in the previous solution, although it already explored it without finding better solution. Because, this strategy may require excessive running time we excluded such explorations from the VND by applying a reduction strategy. The strategy consists in maintaining two sets, *OrCh* and *ExCh*, required for tracking racks that are no checked for possible improvements. The OrCh set is initiated with all racks that are present in the system, while ExCh is initiated with all possible combination of pairs of racks without repetitions. Or-opt *n* and Exchange *n* moves are executed only for members of the appropriate set. Accordingly, when both sets are empty VND ends.

However, each time an improved solution is found those sets are updated and new members are added. In the case when new solution is found in the Or-opt *n* neighborhood structure, the rack on whose allocation Or-opt move resulted in the improvement is added to the *OrCh* set, i.e. $OrCh \leftarrow OrCh \cup \{r\}$, and the VND is restarted. In the case of the improvement due to the Exchange move, beside a pair of racks included in the improvement move (r_1, r_2), we add to the *ExCh* all pairs of racks in which one pair member is r_1 or r_2, i.e. $ExCh \leftarrow ExCh \cup \{\{r_1 \times R \setminus \{r_1\}\}, \{r_2 \times R \setminus \{r_2\}\}\}$. Moreover, for the case of an improvement in the Exchange *n* neighborhood structure OrEx is updated by adding both racks included in the move which resulted with an improved solution, i.e. $OrCh \leftarrow OrCh \setminus \{r_1, r_2\}$.

Exploration of neighborhood structures in a VND can be realized in two possible ways. The first one is more time consuming since it implies that all neighborhood structure solutions are checked and that VND is restarted with the best-found solution as the new x_{loc_opt}. Due to its comprehensiveness this exploration strategy is called the best improvement (BI) VND. On the other hand, there is a strategy called the first improvement (FI) strategy with implies restart of the VND with each founding of a new improved solution, i.e. with an improved x_{loc_opt}. Since in the literature there are no guaranties that an implementation of the BI VND results in better quality of

solutions, and since it usually results with higher running times of the VND compared to the FI VND for solving the considered problem we implemented the FI variant of the VND.

4. Numerical Experiments

In order to test the efficiency of proposed solution methods we tested them on an imaginary typical OP zone layout consisting of 4 aisles and 23 picking location on each aisle side. Nevertheless, due to the existence of building pillars in 10 picking locations the overall number of picking locations, as well as SKUs is 174 (Figure 7). Additional features of the test instances include:

- Expected picker delay times are generated within the interval of 14 and 1200 seconds and are available upon the request to the corresponding author;
- *NoUnSrch* parameter of VNS based algorithms is tested for values of 1,5,10,15,20,25,30,35,40,45 and 50;
- Due to the randomness present in the realization of VNS based algorithms each realization of VNS based algorithms is repeated 10 times.

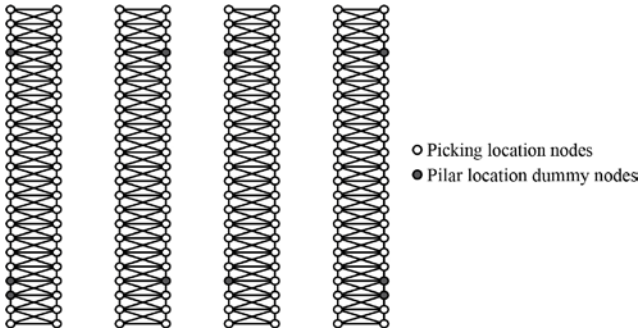

Fig.7.
Testing OP zone Layout

Quadratic mixed integer programming model is solved by Gurobi 8.1 solver while the model is generated by using Gurobi's Python API. All coding for realization of proposed VNS based algorithms is realized in the Python 3.6 programming language. All models are executed by the Intel i7 4712MX CPU with 12GB of RAM and Windows 8.1 as OS.

Before we give overview of obtained results by applied solving approaches, we must emphasize that objective value of the initial solution is 169 586.44s. In the case when the quadratic mixed integer programming model was given an interval of 3600s for solving the problem it found the best solution with the objective function of 166 816.9s. In other words, it reduced the initial solution's objective value for only 1.63%. In the case when available time interval for solving was increased to 24h the best solution obtained with the quadratic mixed integer programming model was 155227.27s, i.e. it improved the initial solution for 8.46%.

Results of the VNS based algorithms are given in the Table 1. in such way that the first column of the table contains value of the parameter NoUnSrch related to the number

of times shaking will be executed without the improvement of the incumbent solution it the VND. The following three columns contain the data related to the quality of obtained solutions by the implementation of the nnVNS algorithm, and the rest three columns for the nVNS algorithm. The second column contains information regarding the average value of obtained solutions given as the percentage average reduction of the value obtained with the QMIP model with 24h limit. The third column depicts the variation of the objective values of obtained solutions, given as a percentage of the obtained objective values. Finally, the third column depicts time requirements of the implemented algorithm.

It is noticeable from the data in Table 1 that nnVNS obtained slightly better results regarding the value of the objective function for all values of the *NoUnSrch*. However, the nVNS obtained the best solution whose objective value was 97468.95s comperd to the nnVNS's best solution with the objective value of 99243.41s. nVNS's best solution was obtained for *NoUnSrch* value of 35 while nnVNS's best solution is obtained for the parameter value of 45.

Regarding the time efficiency of algorithms, it is noticeable that nVNS required significantly shorter period of time for finishing. The difference increases as the value of *NoUnSrch* increases. This implies that the number of VND realizations without an improvement, especially for small values of u, is lower in the case of nVNS, meaning that the nnVNS more depends on the value of the *NoUnSrch* parameter.

Table 1
Results of the nnVNS and nVNS Algorithms

NoUnSrch	nnVNS			nVNS		
	Average Improvement of the QMIP Objective Value [%]	**Standard Deviation of the Objective Function [%]**	**Average Running Time [s]**	**Average Improvement of the QMIP Objective Value [%]**	**Standard Deviation of the Objective Function [%]**	**Average Running Time [s]**
1	24.44	2.67	85.15	23.83	0.93	83.36
5	25.23	3.08	120.24	25.01	4.71	112.76
10	25.89	5.30	161.96	25.20	2.82	130.44
15	26.01	4.64	198.16	25.31	4.02	133.02
20	26.34	5.14	208.18	26.39	4.95	139.45
25	26.64	4.99	227.70	26.35	5.18	158.86
30	27.42	4.43	262.75	26.92	6.31	161.03
35	28.22	6.09	283.72	27.30	5.93	178.86
40	28.71	7.15	294.96	27.21	4.89	179.24
45	28.85	6.41	336.03	28.33	6.33	200.33
50	28.87	5.79	387.67	28.35	6.28	219.05

5. Conclusions

In this research we approached from the tactical level to solving the congestion problem in the order picking problem. Precisely, we considered re-optimization of periodical storage keeping units in such a way that situation which leads to congestion of pickers in ails are minimized. Although, the major improvement in congestion reduction lies in the scheduling of order pickers, allocation of goods over the order picking zone can also contribute the final goal. Two solution approaches that we introduced confirmed that. In the case of the Quadratic Mixed Integer modelling approach, we obtained a solution which was cca. 8% better than the initial one, while in the cases of Variable Neighborhood Search based algorithm approaches, we improved the initial solution for cca. 28%. However, since order picking is very complex operation dependent on numerous aspects of the problem, it offers a wide variety of possible bringing of the problem to the realistic conditions. In that respect, our plans for further work on this problem include incorporation of additional parameters that have an influence on the pickers delay time at picking locations. Such parameters are frequencies of SKUs' picking, weight of the picked units, etc. Besides that, since complexity of the problem is very high, especially for warehouses of several hundreds of locations in the order picking zones further improvement of the heuristic based solution methods is also the issue for the further work. Finally, based on the strong operational aspect of the order picking process it is reasonable that some kind of simulation-based models, for comparing effects of order picking process with and without implemented allocation optimization, should be also realized.

Acknowledgement

This work was supported by the Ministry of education, science and technological development of the Government of the Republic of Serbia through the grant number TR36006, as well as through the project of bilateral cooperation between the Republic of Serbia and the Republic of Slovakia in the period 2019-2021, grant number 337-00-107/2019-09/06.

References

Bataineh, M.; Khasawneh, M. 2016. The Effect of Picking Congestion in Manual Order Picking Environments. In *Proceedings of the 5th Annual Conference of the Society for Industrial and System Engineering*, San Francisco, USA, 256-263.

Dallari, F.; Marchet, G.; Melacini, M. 2009. Design of order picking system, *The International Journal of Advanced Manufacturing Technology* 42: 1–12.

de Koster, R.; Le-Duc, T.; Roodbergen, K. J. 2007. Design and control of warehouse order picking: A literature review, *European Journal of Operational Research* 182(2): 481 – 501.

Djurdjević, D. 2019. *Order picking technologies for piecewise goods*. Belgrade: University of Belgrade – Faculty of Transport and Traffic Engineering, CD edition (In Serbian).

Frazelle, E. 2001. *World-Class Warehousing and Material Handling*. NewYork: McGraw-Hill. 256p.

Gu, J.; Goetschalckx, M.; McGinnis, L. F. 2010. Research on warehouse design and performance evaluation: a comprehensive review, *European Journal of Operational Research* 203: 539–549.

Hansen, P.; Mladenović, N. 2014. Variable Neighborhood Search. In: *Burke E., Kendall G. (eds) Search Methodologies*. Springer, Boston, 313-337.

Hansen, P.; Mladenović, N. 2001. Variable neighborhood search: Principles and applications, *European Journal of Operational Research* 130: 449–467.

Hompel, M.; Schmidt, T. 2007. *Warehouse Management: Automation and Organisation of Warehouse and Order Picking Systems*. Berlin: Springer. 356p.

Mladenović, N.; Hansen, P. 1997. Variable neighborhood search, *Computers and Operations Research* 24(11): 1097–1100.

van Gils, T.; Ramaekers, K.; Carisa, A.; de Koster, R. 2018. Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review, *European Journal of Operational Research* 267(1): 1-15.

Wascher, G. 2004. Order picking: a survey of planning problems and methods. In: *Dyckhoff, H., Lackes, R., Reese, J. (eds) Supply Chain Management and Reverse Logistics*. Springer, 324 – 370.