Routing order pickers in warehouses considering congestion and aisle width

Alireza Eydi[1*], Hanif Mohagheghi[2], Seyed Ali Ghasemi-Nezhad[3]

[1]Associate professor in the Faculty of Engineering, University of Kurdistan, Sanandaj, Iran
Alireza.eydi@uok.ac.ir, Mobile numbers: +989188622330
[2,3]MSc of Industrial Engineering, University of Kurdistan, Sanandaj, Iran
sa.ghaseminezhad@gmail.com, hanif_mohagheghi@yahoo.com

Abstract
Due to the importance of routing order pickers, there has been extensive research in the area of routing in warehouses. Still, there are some prominent factors that should receive more attention, as they may provide unsatisfactory services and incur considerable operational costs if ignored. In real-world applications, warehouse configuration, width of aisles, and controlling the vehicle congestion in the aisles greatly influence the efficiency of the routing process. Therefore, this paper proposes a mixed-integer programming model. The model aims to minimize maximum delivery time by finding the shortest pickup and delivery routes of all goods for all vehicles. Since the problem is NP-hard, a Simulated Annealing metaheuristic approach is designed to solve the model in large-size problems. This research contributes to picker routing literature by considering dynamic congestion, narrow and wide aisles, and pickup times and proposing a metaheuristic algorithm. The validation and efficiency of our proposed model are proved by solving some various generated benchmark problems. In summary, the developed route planning mathematical model works effectively for any two-dimensional rectangular layout, and the collision prevention constraints are incorporated in the mathematical model.

Keywords: Picker routing; warehouse; aisle width; congestion; simulated annealing

Nomenclature: The following abbreviations are frequently used in this paper:
TSP: Travelling Salesman Problem
MHS: Material Handling System
SA: Simulated Annealing

1. INTRODUCTION
Warehouses play an important role in supply chain management systems. Because of the rapid growth in market competition, they require better material handling systems that can keep up with everyday growing demands. Receiving, storage, order picking, sorting, packing, and delivery of goods are the main activities done in a warehouse; but what connects all these activities together is the routing activity. The routing activity takes about half of the typical distribution time of an order picker [1,2]. Vehicle movement or travel is a loss and does not add value. The travel time is an increasing function of travel distance [3]. Therefore, travel distance is often considered the main objective in the warehouse optimization models. The warehouse routing problem deals with the sequence of the items on the pick list and determines which vehicle should pick and deliver which item. The warehouse routing problem is a particular case of the Travelling Salesman Problem (TSP) known as Steiner Travelling Salesman Problem (STSP). The main difference between the two problems is that not all the nodes (pick locations) have to be visited in a single tour in STSP and that some nodes can be visited more than once [4]. The STSP problem is generally not solvable in polynomial time (unless P = NP). Therefore, the application of heuristic or metaheuristic approaches is required to solve the routing problem in a more reasonable time.
The routing process is significantly influenced by the storage policy. There are mainly five routine storage policies in warehouses: random storage, class-based storage, dedicated storage, closest open location storage, and full turnover storage

*Corresponding Author.Tel: +988733660073; fax: +988733660073
Faculty of Engineering, University of Kurdistan, Pasdaran Blvd., Post box no.: 416, Sanandaj, Iran

[4]. All the storage policies have their own advantages and disadvantages regarding the routing problem, and based on the nature of the inventory control system and warehouse configuration, each policy can be adopted to store items in racks [5].
The order picking process consists of classifying and scheduling orders so that the operators could pick all the demanded items as quick and convenient as possible subject to resource constraints such as labor and machines (see [6]). It is important to know whether the picking activity is done by machines or humans. The majority of warehouses apply picker-to-part systems, where the order picker (humans) walks or drives along the aisles to pick items [7]. Order picking systems are mainly divided into three groups: single-order picking, zone picking, and batch picking [8]. When orders are large, each order can be picked individually. However, it is possible to reduce travel times by picking a set of orders in a single tour when orders are small. This way of picking orders is known as batch picking. Gademann and Van de Velde [9] demonstrated that the order-batching problem with the objective function of minimizing the total travel time is NP-hard when the number of orders per batch is greater than 2. Many studies have focused on developing heuristic methods to solve the order-batching problem. As a result, two popular order-batching heuristics were proposed: seed algorithm and saving algorithm. Chen and Wu [10] developed a new model for proximity batching based on binary integer programming to maximize the total association of batches using the data mining approach. Another type of order-batching is the time-window order batching where the orders arriving during the same time interval (time window) are grouped as a batch. Le-Duc and De koster [11] considered variable time-window order batching with stochastic order arrivals for the manual order picking. The results from the simulation experience represent high accuracy and simplicity in practice. In the zone picking method, the order picking area can be divided into several zones where each order picker is assigned to a specified zone. Little literature on this area is available. Parikh [12] addressed two key issues in the order-picking system design: selection between batch and zone order picking strategy, and configuration of storage system. Peterson [13] showed that the average travel distance within each zone depends on the number of aisles per zone, aisle length, and number of items on pick list. Several heuristic algorithms for assigning stock keeping units (SKUs) to zones can be found in the studies of Jane [14] and Jewkes et al. [15].
Due to the complexity of warehouse routing problem and the lack of optimal solutions in some special cases, many heuristics are proposed to solve the problem [9]. S-shape heuristic, return mode, midpoint method, and largest gap method are among the most classical heuristics proposed in the literature for single-block warehouses (see [16]). Petersen [5] performed a number of numerical experiments to compare the routing methods in single-block warehouses. Roodbergen and De Koster [17] compared the heuristics with the optimal solution in 80 warehouse cases including the aisles varying between 7 and 15, cross aisles varying between 2 and 11, and the pick-list size between 10 and 30. The results represented that a combination of all the classical heuristics gives the best solutions for about 93% of the warehouses.
There are few researchers who have focused on the warehouse routing problem while considering the vehicle congestion, and those who have considered the congestion during the routing process assume that all the aisles in the warehouse have equal width to simplify their model. Congestion is known as a situation in a warehouse where vehicles (order pickers) cannot move by their defined speed or the movement of vehicles at the same time which will lead to collision. Therefore, the vehicles should wait for other vehicles to move away or should choose different routes when they happen to face each other. The major factors that provide congestion in a warehouse are intersections (where aisles meet cross aisles), number of vehicles using the same aisle in the same time interval, pick-up and drop-off operations, location and density of items on pick list, and aisle width [18]. Among the mentioned factors, the aisle width has a prominent influence on the congested warehouses, especially in narrow aisles when two or more vehicles attempt to pass at the same time and one has to move back to allow the other one to pass.
In general, congestion cannot be predicted in advance. Although researchers have determined different criteria to predict the congestion so that it is reduced as much as possible in practice, the formal studies in this area are still limited. The following review includes the formal discussions of traffic congestion in the manufacturing facilities. Smith and Li [19] addressed queuing the congestion and modeled the material handling aisle network as a single server focusing on minimizing the total number of customers. Chiang et al. [20] pioneered developing the Quadratic Assignment Problem (QAP) to model workflow interference. It was only included between the pairs of flow. The unit interference cost of having cross flow was quantified as 1 and the opposed flow was quantified by parameter ω. They later extended their work by adding distance objective to work flow interference objective (see [21]). Herrmann et al. [22] studied the path design problem in material handling as a fixed-charge capacitated network design model. The model limited the flow through a link to a bound to help reduce the congestion. Bakkalbasi [23] quantified the congestion effect as a number called congestion or traffic factor. He argued that the blocking time caused by congestion should be considered when determining the fleet size. Vosniakos and Davies [24] introduced an index that was the percentage of the time the track segments were blocked because of vehicle obstruction. The index defined the network congestion. Kim and Tanchocho [25] also measured the congestion by defining an index as the actual travel time divided by shortest travel time if there was no congestion. Following their idea, Beamon [26] developed a method for measuring the congestion in Material Handling System (MHS). Zhang et al. [18] discussed workflow congestion in MHS and modeled the rerouting problem as a multi-commodity flow problem. Also, they developed a simulation model to explicitly consider various workflow interruptions. Pan and Wu [27]

studied an approximation method based on the queuing network to find the throughput time and waiting time of an order picking system with multiple pickers and aisle congestions. Hong at al. [28] developed a model by integrating the batch sequence selection with the routing and batching decisions to control the picker blocking. Kim and Lee [29] considered a dynamic inbound ordering and shipment scheduling problem for multiple products that are transported from a supplier to a warehouse by common freight containers. Gokhan Ozden [30] developed a warehouse layout optimization system that calculates the routing distances and performs heuristic optimization over a comprehensive set of layout design parameters. For multi-block layouts, Scholz and Wascher [31] integrated different routing algorithms into an iterated local search approach for batching to demonstrate the benefits gained from solving the order batching and picker routing problems in a more integrated way. Chen et al. [32] developed a routing algorithm to address the newly observed limitations imposed by ultra-narrow aisles and access restriction. Hojaghania et al. [33] proposed a novel MINLP for on-line batching to improve the performance of warehouse, which in turn results in reducing the response time and idle time. The mentioned model was solved using two algorithms of bee colony and ant colony. Tajima et al. [34] modeled an order picking operation in which two or more pickers are operating simultaneously. In this research, the behavior of the pickers was modeled using multi-agent systems. Zuniga et al. [35] proposed a model that improves the order picking by making simultaneous decisions on the storage location assignment and the picker-routing problem considering the precedence constraints based on the product. Cano et al. [36] formulated the STSP models considering multiple pickers, heterogeneous picking vehicles, multiple objectives, and due windows. As more companies seek cost savings in their warehouses, the problem of routing order pickers in a warehouse becomes a bigger concern for the whole supply chain management, and developing more efficient models will be a forward step towards better customer satisfaction. So far, many papers have studied order picking processes to some extent, but there is still a gap between practice and academic research.

There are very few papers that include aisle configuration and vehicle congestion simultaneously within the warehouse routing problem. The Pai's work [37] is one of the very good attempts to propose models for such problems and inspired us to write this paper and develop a novel model by solving the approaches for routing order pickers considering the dynamic form (time-integrated) of congestion, aisle width and pick up/delivery time of goods. In spite of the fact that the Pai's work provided the primary motivation, it needed some major modifications to make the whole model more practicable. The motivation for this work is derived from the insight into two considerations. The primary motivation is the lack of a comprehensive methodology for the collision prevention in warehouse, while the secondary motivation is ignoring narrow and wide aisles in a route planning model. To plan picker routing in the practical sizes, we also design a simulated annealing process and adjusted it to meet the specific features of the problem. Finally, we make the comparison and perform benefit analysis on the proposed algorithm. In summary, this paper contributes to picker routing literature in the following respects: (a) A mixed-integer programming model of the problem is developed considering the dynamic congestion, narrow and wide aisles, and pickup times; (b) A simulated annealing algorithm is proposed to solve the problem; (c) A validation approach is proposed for the problem.

The rest of this paper is organized as follows: In section 2, the model formulation is presented. Section 3 discusses the solving approaches used for the model. In section 4, the computational results are presented and analyzed. Section 5 discusses the conclusions and future research for the proposed model.

2. MODELING

The following assumptions are made to develop the model proposed for the problem:

- The warehouse configuration is a two-dimensional rectangle, and the assumed warehouse dimension is small (like manufacturing material handling problem).
- Each rack within the warehouse can be used to store only one item.
- The rack locations and the number of locations are fixed and identical.
- There are a fixed number of both narrow and wide aisles in the warehouse.
- The number of material handling vehicles (pickers) are known and fixed.
- All vehicles are of the same type and are capable of picking up and dropping off all items.
- All vehicles can move in both directions in all aisles, and also vehicles cannot move in cater-corner directions in wide aisles.
- The information on requested demands, number of items on pick list, and location of the items is known in advance.
- The capacity of depot (input/output point) is assumed to be unlimited.
- The waiting time for picking orders is considered as one time unit for all items on the pick list.
- The queuing process is not considered inside the model.
- In mathematical model, a discrete time horizon is considered.

2.1. Model formulation

In this section, the model formulation is presented. Before that, the set of indices, parameters, and decision variables used in the model are described.

Sets of Indices:

i: Index of grid set (a grid network represents the warehouse) (i=1,…,I)
j: Index of vehicle set (j=1,…,J)
t: Index of time sets (t=1,…,T)
S_i: If grid *i* (*i* is an element of set S_i) belongs to an aisle, it is the set of grids to which the vehicles can move in the next time instant. If grid *i* belongs to a rack, it is the grid adjacent to the rack grid that the vehicle should go to and pause there for the defined pickup waiting time in order to pick up the target item in grid *i*.
O: Set of rack grids in which the target items to be picked exist.
Dep: Delivery grid for the picked up items (subset of set *I*).

Parameters:

M: A very large positive number

ε : A very small positive number

c_j : Capacity of vehicle *j*

LT: Waiting time needed to pick up an item

The final values of *M* and ε are chosen by trial and error method during the coding process. The values are offered based on the results from similar works and also the time required to solve the problem.

Decision Variables:

P_{ijt}=1 if vehicle *j* meets grid *i* at time *t*; otherwise P_{ijt}=0.

L_{ijt}=1 if vehicle *j* picks up an item at grid *i* at time *t*; otherwise L_{ijt}=0

D_{ijt}=1 if vehicle *j* drops off an item picked up at grid *i* in the depot at time *t*; otherwise D_{ijt}=0.

f_{ijt}: Delivery time of item at grid *i* to the depot by vehicle *j* at time *t*.

f^1_{ijt}: Pickup time of item at grid *i* by vehicle *j* at time *t*.

The model is formulated as follows:

$$Objective: MinimizeW$$

$$W \geq \sum_{j=1}^{J}\sum_{t=1}^{T} f_{ijt} \qquad \forall i \in O \tag{1}$$

$$f_{ijt} \leq M D_{ijt} \qquad \forall i \in O, \forall j, \forall t \tag{2}$$

$$f_{ijt} \leq t \qquad \forall i \in O, \forall j, \forall t \tag{3}$$

$$f_{ijt} \geq t - M(1 - D_{ijt}) \qquad \forall i \in O, \forall j, \forall t \tag{4}$$

$$f^1_{ijt} \leq M L_{ijt} \qquad \forall i \in O, \forall j, \forall t \tag{5}$$

$$f^1_{ijt} \leq t \qquad \forall i \in O, \forall j, \forall t \tag{6}$$

$$f^1_{ijt} \geq t - M(1 - L_{ijt}) \qquad \forall i \in O, \forall j, \forall t \tag{7}$$

$$\sum_{t=1}^{T} f_{ijt} - \varepsilon \geq \sum_{t=1}^{T} f^{1}_{ijt} \qquad \forall i \in O, \forall j, \forall t \tag{8}$$

$$\sum_{j=1}^{J} P_{ijt} \leq 1 \qquad \forall i, \forall t \tag{9}$$

$$P_{kjt} + P_{ilt} + P_{ij(t+1)} + P_{kl(t+1)} \leq 3 \quad \forall i, \forall j, \forall t, k \in S_i, l \in J, l \neq j \tag{10}$$

$$\sum_{l \in S_i} \sum_{j=1}^{J} P_{ljt} \leq 3 \qquad \forall i, \forall t \tag{11}$$

$$\sum_{k \in S_i} P_{kj(t+1)} - P_{ijt} \geq 0 \qquad \forall i, \forall j, \forall t \tag{12}$$

$$L_{ijt} \leq \sum_{k \in S_i}^{I} P_{kjt} \qquad \forall i \in O, \forall j, \forall t \tag{13}$$

$$D_{ijt} \leq P_{(Dep)jt} \qquad \forall i \in O, \forall j, \forall t \tag{14}$$

$$\sum_{i=1}^{I} P_{ijt} = 1 \qquad \forall j, \forall t \tag{15}$$

$$\sum_{i \in O} \sum_{k=1}^{t} D_{ijk} + C_j \geq \sum_{i \in O} \sum_{k=1}^{t} L_{ijk} \quad \forall j, \forall t \tag{16}$$

$$\sum_{j=1}^{J} \sum_{t=1}^{T} L_{ijt} = 1 \quad i \in O \tag{17}$$

$$\sum_{j=1}^{J} \sum_{t=1}^{T} D_{ijt} = 1 \quad i \in O \tag{18}$$

$$\sum_{b=0}^{LT} P_{kj(t+b)} \geq L_{kjt} * (LT+1) \qquad \forall k \in O, \forall j, \forall t \tag{19}$$

$$P_{ijt}, L_{ijt}, D_{ijt} = 0 \text{ or } 1 \; ; \; f_{ijt}, f^{1}_{ijt} \geq 0 \tag{20}$$

In the model above, **Equation related to the objective function:**
Constraint (1) ensures the minimization of maximum delivery times of all goods. In other words, the objective function is adopted with the following formulation: $Minimize W = \max_{i \in O} \{\sum_{t=1}^{J} \sum_{t=1}^{T} f_{ijt}\}$

Constraints related to pickup and delivery time:
Constraints (2), (3) and (4) are a combination of the constraints ensuring that $f_{ijt}=t$ if item *i* picked up by vehicle *j* is delivered to the depot at time *t*; otherwise, $f_{ijt}=0$. Constraints (5), (6) and (7) are also a combination of the constraints ensuring that $f^{1}_{ijt}=t$ if item *i* is picked up by vehicle *j* at time *t*; otherwise, $f^{1}_{ijt}=0$. Constraint (8) states that the delivered item is definitely picked up in advance by the same vehicle. In equation (8), to change the greater sign into equal or greater sign, parameter ε is subtracted.

Constraints related to vehicles (congestion control and collision avoidance):
Constraint (9) states that each grid is occupied by up to one vehicle at any time. Constraint (10) requires that if two vehicles occupy adjacent grids at any time instant, they will not switch their positions in the next time instant. Constraint (11) restricts number of adjacent vehicles in wide aisles to 3 so that the collision risk of vehicles is limited as much as possible.

Constraint related to movement feasibility:
Constraint (12) represents the feasibility and continuity of the problem. In each successive time instant, a vehicle should either stay at its current position or move to the next feasible adjacent grid.

Constraints related to vehicle positioning:

Constraint (13) ensures that when item *i* is picked up by vehicle *j*, vehicle *j* is definitely in a grid in the aisle adjacent to the grid in which the item should be picked. Constraint (14) ensures the availability of vehicle *j* in the depot when dropping off item *i*. Constraint (15) states that all grids are occupied by up to one vehicle at any time instant.

Constraint related to vehicle capacity:

Constraint (16) requires that number of items picked up by any vehicle should not violate its capacity.

Constraints related to pickup and delivery:

Constraint (17) states that an item is picked up exactly once by one vehicle. Constraint (18) states that a previously picked up item is dropped off exactly once in the depot. Constraint (19) represents the waiting time for picking up an item. It states that the vehicle will stay for a defined pickup waiting time at the grid adjacent to the grid whose item should be picked.

Finally, constraint (20) represents the domain of the decision variables.

2.2. Graphical representation and sample generation

Due to unavailability of benchmark problems in the literature, some sample problems are generated to include the diversity of samples. Figure 1 depicts the configuration of a sample problem.

In Figure (1), the warehouse is divided into 36 grids; each grid representing aisles (in light blue), racks (in dark blue) or depot (in red). It has one wide aisle and one narrow aisle. In the narrow aisle, the order picker vehicle is able to retrieve items from both sides of the aisle without changing the position, but in the wide aisle, the vehicle needs to move (physically) from one side to the other to pick items on both sides of the aisle. Also, there is a cross aisle connecting the wide and narrow aisles together. There are three other columns consisting of the racks into which the items are stored. The vehicles are not supposed to move to the grids named as racks and can only move in the grids named as aisle and depot grid. To pick up an item in a rack, a vehicle should move to its adjacent grid (See S_i). After a vehicle reached the desirable grid to pick up an item, it should wait as long as its waiting time. For example, if the vehicle in grid 12 wants to pick up the item in grid 3, it should move to grid 10 and wait as long as the determined waiting time needed to pick up the item. It should then move to grid 31 (depot) and deliver the picked item. Furthermore, this paper assumes that each vehicle has to drop off the first item it picks up and then continue to go for picking up another item. In other words, the capacity of vehicles is considered as one item for each vehicle in the model so that the movement of the vehicles in the warehouse is maximized. This means that one vehicle cannot pick up all the items on its way to the depot without letting other vehicles to move for obtaining the items on the pick list. The intensification of vehicle movements increases the chance of blocking or colliding the vehicles, and the dynamic vehicle movement is required to demonstrate that our proposed model is undoubtedly a collision-free model.

In the next section, the computational results derived from running the model are given by some sample problems.

2.3. Model experiments

To evaluate the model, some experimental problems were tested on the model. Since there is no benchmark problems in the literature, some random problems were generated. For generating the samples, we tried to have different warehouse dimensions, different number of aisles and vehicles, different types of aisles, and different number of items on the pick list. The proposed model is solved using CPLEX solver by GAMS23.8.2 software on a computer equipped with Core i3 3.1 GHz CPU and 4 GB RAM running a Windows 7 operating system. The results are shown in Table 1.

In all the generated sample problems, the width of wide aisles is twice that of narrow aisles. Therefore, only one vehicle can thoroughly pass along a narrow aisle if another vehicle is coming in the opposite direction; but at least two vehicles can pass along each other in wide aisles. In most of the problems, the initial position of vehicle is considered in the depot when there is one vehicle, and for more than one vehicle, the grids next to the depot are occupied with other vehicles, respectively, for the initial position of other vehicles. Empirically, the maximum

allowed time (upper bound for the algorithm or T_{max}) is also considered as twice the grid size. Figure 2 presents the warehouse configuration of generated problems 7, 8 and 9.

As is seen in the results, there is no certain way to identify the parameters influencing the problem complexity and solution time. However, what can be found from the results is that the solution time is a factor dependent on grid size, number of vehicles, initial position of vehicles (e.g., problems 11 and 12), and number and location of items on the pick list. The only certain factor that definitely increases the solution time is the problem size. As the time required to solve instances of the problem grows as non-polynomial with the size of the instances, the problem is called NP-hard. It requires heuristic and metaheuristic approaches for solving the proposed model to gain good solutions in a reasonable amount of time. A metaheuristic approach is suggested as a solution in the next section. All the generated samples solved by the exact algorithm in this section will be solved by the proposed metaheuristic approach in the next section based on the SA, and the results will be discussed and analyzed.

3. PROPOSED SOLUTION METHOD

Although the exact methods are capable of solving problem instances of small and medium sizes, in large-sized problems, the heuristic or metaheuristic solving approach is required for creating good solutions with respect to run times. Simulated annealing algorithm, as a solution approach in combinatorial optimization, is suitable to avoid local optimum solutions by accepting the worse solutions, ensuring that the solutions obtained are not stuck in local optima. Furthermore, SA is used to search the solution space in an effective manner. Simplicity of implementation, good adaptability of the algorithm to our model, and the way to represent solution structure were among the most important reasons that we chose the SA algorithm among all the available metaheuristics in the literature. Furthermore, SA is a local search algorithm and can work with the models with many constraints, which are a potential for our model. Metropolis et al. [38] proposed an algorithm for analyzing the changes in the temperature of solid materials. Later, Kirkpatrick et al. [39] and Cerny [40] used the Metropolis's idea of cooling the materials gradually for minimizing the objective (cost) by applying the simulation approach. Simulated annealing is a robust general technique, which is widely and successfully used to solve NP-hard problems [41].In the following subsections, we will show how this process (proposed SA) works.

3.1. SA algorithm process

The following steps are needed to reach the final solutions in SA algorithm:

- Generation of initial solution
- Updating temperature
- Determining length of Markov chain (SA is modeled as a sequence of solutions)
- Generating neighborhood solutions
- Fitness evaluation and assigning penalties for infeasible solutions
- Stopping rule to reach final solution

The overall structure of coding Simulated Annealing is described below:

- Initial temperature (T_0) is selected.
- Solution S_0 is chosen randomly from the solution space as the initial solution.
- $\Delta = F(S_1) - F(S_0)$ is calculated where $F(S_1)$ is the new solution and $F(S_0)$ is the initial solution.
- If $\Delta<0$, then $S_1=S_0$.
- If $\Delta\geq0$, then $0<x<1$ is randomly generated and if $x<exp\ (\frac{-\Delta}{b.T})$ where x is a random number, b is the Boltzmann constant and T is the current temperature.
- The counter number of iterations in the current temperature is added by one number.
- If the counter number does not exceed Markov chain length, the previous steps are repeated; otherwise, the next step is applied.
- Temperature is updated by $T_{k+1} = \alpha T_k$. Parameter α is called the cooling rate.
- The best obtained solution is considered as S_0.

3.2. Representation of solution

An effective coding process has a significant effect on the performance of SA algorithm. We employed a multi-row structure to represent the initial solution and final solution obtained by the algorithm. The first J rows represent the routes of all vehicles moving to and the $J+1$th row represents the assignment of vehicles to the target items of the pick list. Take problem 8 depicted schematically in Figure 2 as a sample to help elaborate on the representation of routing and picking solutions. In this problem, there are two vehicles and three items on the pick list. The initial locations of vehicles are grids 24 and 23. As shown in Figure 3, the first two rows depict the initial routes of first and second vehicles, respectively. The third row states that the first vehicle (shown by 1001) should pick up one target item located in grid 13. As a result, it should move to grid 14 that is the grid adjacent to grid 13. The third row also states that the second vehicle (shown by 1002) should pick up two target items on grids 4 and 12. Therefore, it should move to grids 5 and 11, respectively, to pick up each one. All of the other cells in the third row are filled by zeroes. The number of integers in each row is assumed to be as much as parameter T_{max} in the model which guarantees that the final solution does not exceed the upper bound considered for each problem in the model. The final solution of the algorithm has the same structure of the initial solution shown in Figure 3.

Set S_i should also be defined for all grids so that warehouse configuration (aisles, racks and depot) is defined in the C++ language, because the constraints are very dependent on the warehouse configuration. Set S_i is defined by a matrix having six columns for all problems where the number of rows are equal to problem size. Each row represents a grid of its own rank. The last column in each row represents the maximum number of allowed moves in the grid. When all allowed moves for the particular grid are filled, the rest of the cells are filled with zeroes. Figure 4 presents set S_i for problem 8 in which the initial routing solution was discussed before in this section. The problem size in this problem is 24. Therefore, the number of rows is 24. Each row shows its relative grid number in the warehouse layout. For instance, the ninth row highlighted in orange shows grid 9 and the allowed moves from the grid. As can be seen from Figure 2, the vehicle can move up to four allowed directions from the grid and is allowed to go to grid 3, 8 or 15 or stay in its current grid, namely grid 9. The maximum allowed moves from the grid are shown in the sixth column and the location of allowed moves are shown in the first columns and the rest is filled with zero. The rows whose total cells are filled with zeroes represent the racks with no items to be picked from, and the rows whose sixth columns are filled with number 1 are those where the vehicle will pick up the item on its adjacent grid when reaching the grid.

3.3. Tuning SA parameters

The performance of Simulated Annealing algorithm is very sensitive to parameter settings. Therefore, it is very necessary to tune the parameters to their best values. We suggested values 100, 200 and 400 for initial temperature and values 0.001, 0.01 and 0.1 for cooling temperature. For cooling rate, values 0.99, 0.98 and 0.97 were evaluated. For number of iterations in each temperature, values 3000n, 5000n and 700n were used where n is the grid size.
In order to find the most suitable parameters for our problem, a subset of random generated problems is chosen and the following process is applied:

- To find the best parameter for the initial temperature, all other parameters are set to their minimum value and initial temperature values 100, 200 and 400 are evaluated for the problem.
- Then, the best value of initial temperature that results in better solutions is fixed for the rest of the parameter evaluation.
- Next, the same process is done for the cooling temperature, while the initial temperature value is set from previous step and other parameters are again set to their minimum levels.
- The best value of cooling temperature that results in better solutions is set as cooling parameter.
- This continues until all parameters are set.

After evaluating various combinations of the parameters for the set of problems, value 200 for initial temperature, value 0.001 for cooling temperature, value 0.98 as the cooling rate, and 5000n as the number of iterations in each temperature were set for the parameters.

3.4. Neighborhood structure generation

In order to find better solutions and make sure that the algorithm searches the solution space as much as possible so that it does not skip possible good solutions, two different neighborhood structures are proposed: neighborhood structure for vehicle routing, and swapping neighborhood structure. Before describing each structure, it should be noted that both structures concern that the routing solutions for each vehicle should be collision-free. This means

that the data in each column should never be the same at any time instant. It is also noted that each vehicle can carry only one item at a time, which means that one vehicle has to deliver the first picked up item before being able to pick up another target.

3.4.1. Neighborhood structure for routing

This structure helps to improve the vehicle routing solution. It aims to find shortest routes of picking up and delivering the goods for vehicles. The structure works as follows:

Step 1: A vehicle is chosen randomly.
Step 2: One of the grids the vehicle passes in step 1 is chosen.
Step 3: A new random route is generated from the grid.

Figure 5 shows how neighborhood structure works for problem #8. There are two vehicles in this problem (two rows) and three items to be picked (grids 14, 11 and 5 in the third row). The upper figure is a routing solution for the problem stating that the first vehicle should pick up and deliver one target and the second one has two targets to pick up and deliver. As mentioned above, in this problem, the second vehicle is chosen randomly, and grid 24 is chosen randomly in its initial routing. As represented in the figure below, a new routing structure is then generated from the grid.

3.4.2. Swapping neighborhood structure

This structure aims to change the assignment of goods among vehicles. Due to the initial location of any vehicle at any time instance, it is important to change the assignment of goods among vehicles. This helps find better solutions by searching all kinds of solution alternatives for assigning goods to vehicles, because at any time instance, the vehicles have different distances from different goods. This structure works as follows:

Step 1: A random number equal or greater than 1 and equal or smaller than the summation of goods and vehicles is chosen. The chosen number is equal to i ($1 \leq i \leq$number of vehicles + number of target items).
Step 2: Another random number is chosen by the same rule as step 1. The chosen number is equal to j ($1 \leq j \leq$number of vehicles + number of target items).
Step 3: The numbers in grids i and j are swapped.
Step 4: This continues until all $i=j$ cases are produced.

Figure 6 presents this structure for problem #8. There are 2 vehicles and 3 target items in this problem. Two numbers are produced between 1 and 2+3 and are named as i and j, respectively. In this problem, i=2and j=4. Swapping grids 2 and 4 in the third row changes the assignment of items 14 and 11 to different vehicles. Totally, this structure either changes the assignment of two items with each other or changes the assignment of vehicles with items. In general, no possible assignment is ignored by this structure.

3.5. Initial solution generation

Like other metaheuristics, SA also needs an initial solution in its initial temperature, and the more accurate the solution, the faster the algorithm in finding better solutions.

Step 1: One of the target items on the pick list is selected randomly.
Step 2: One of the vehicles is selected randomly and is assigned to pick up the item chosen in step 1.
Step 3: A feasible routing solution is made based on set S_i and location of other vehicles using the greedy approach.
Step 4: After reaching the desired grid to pick up the item, the vehicle is paused as long as the defined waiting time in order to pick up the target item.
Step 5: Another feasible routing solution is made to the depot for the vehicle based on set S_i and location of other vehicles using the greedy approach.
Step 6: The above procedure is repeated for the next item on the pick list until all items are delivered to the depot.

3.6. Fitness function evaluation

The fitness function objective of the proposed SA algorithm in the model is to minimize the maximum delivery time of all goods for all vehicles. In this paper, in order to avoid infeasible solutions, some penalties are included in the fitness function of SA algorithm. The penalties used in the algorithm are as follows:

- If an item on the pick list is never picked up.
- If an item is picked up but is not delivered to the depot.
- If any vehicle carries items more than its capacity (unit capacity).
- If two or more vehicles occupy the same grid at the same time instant (collision).

If any of the mentioned cases happens, 100 time units are added to the objective function. Adding such a penalty makes sure that none of the above will ever be considered in the solution.

3.7. stopping criterion

There are some routine methods for stopping criterion in SA algorithm, such as: setting limited number of iterations until a better solution is obtained, and cooling temperature schedule. Generally, the cooling schedule plays an important role in the quality of solutions in SA algorithm. Therefore, among the mentioned stopping criteria, we used cooling temperature schedule.

4. COMPUTATIONAL RESULTS

The simulated annealing algorithm was coded in C++ language and was run using the same personal computer by which the exact algorithm was previously run. The entire previous 20 generated problems were solved by our proposed SA algorithm; each one was run 20 times so that the average solution, worst solution, and best solution are taken for every problem. The results are given in Table 2.

4.1. Comparison between exact approach and SA algorithm

In this section, the exact results obtained from GAMS software are compared with our proposed simulated annealing results coded in C++ language. The accuracy of the results and the time each algorithm reaches its best result are analyzed. Table 3 presents the results of exact algorithm and SA algorithm altogether. It also presents the solution times for each problem. The optimality gap is calculated by formula $\frac{(BP-BF)*100}{\mathrm{BF}}$ where BF is the result value of exact algorithm and BP is the result value of SA algorithm.

The gap results show that the SA metaheuristic approach works quite well for most of the problems and some problems are totally gap-free. Furthermore, those problems whose gaps are not equal to zero have also a reasonable amount of gap, which generally happens in such problems. The optimality gap results represent the efficiency of the proposed SA algorithm for our model.

In Charts (1) and (2), we compare both the results and solution times between exact and SA algorithm, respectively. It is undoubtedly accepted that exact solutions outweigh SA solutions, but as seen in Chart (1) and also as optimality gap states, SA approach produces very good solutions in a reasonable amount of time. Chart (2) presents the major time savings obtained by SA algorithm in comparison with exact algorithm (Chart (2) uses a logarithmic scale). It can be found that in larger-sized problems, the exact algorithm is practically useless, because it is a very long time to reach a solution and, therefore, SA algorithm should be used instead.

5. CONCLUSIONS

The problem of routing order pickers in a warehouse has been always a big concern for the whole supply chain management, and developing more efficient models will be a forward step towards better customer satisfaction. The proposed problem consists of two-dimensional rectangular warehouse, assuming the small warehouse dimension. In this paper, a mixed-integer linear programming model with the objective of minimizing the maximum delivery time

of all goods was proposed. The model worked by finding the shortest routes for the pickup and delivery of all goods. In addition, two important factors, aisle width and congestion, were considered in the routing process. The developed model is able to answer two important questions about picker routing in warehouse: 1) Does the proposed routing method work effectively for any type of warehouse layout? and 2) Does the proposed model prevent vehicles from colliding during the trip?

To evaluate the model, we solved it by GAMS software for some of the generated instances. The results proved its rapid growth as problem size increased and showed that its solving time increases as non-polynomial by problem size expansion. Therefore, the picker routing problems are considered as NP-Hard problems. In order to obtain faster solutions in a reasonable amount of time for large-size problems, we developed a meta-heuristic method based on SA algorithm. The results were compared to the optimal solutions and they proved to improve performance of the proposed algorithm. The other point is that in realistic warehouse situation, the orders arrive continuously over time and the problem needs to be solved using a rolling horizon framework. For further work, the following ideas can be studied by researchers that tend to develop this work:

- Applying the proposed model on a real case study and evaluating the efficiency of the model and its solving approach.
- Solving the model in a multi-depot warehouse.
- Studying the model where each rack can be used to store multiple items.
- Combining the routing method with storing method, especially the zoning storage method for an inventory programming model.
- Developing the model using various vehicles and also various vehicle capacities.
- Solving the model with other heuristics and metaheuristics and comparing the results with the ones used in this paper. In terms of solution approaches, the interested readers can refer to Modiri-Delshad et al. [42], Modiri-Delshad et al. [43], Pourdaryaei et al.[44], and Hlal et al. [45].
- Studying order picking problem under online routing schedule.
- Modifying the model in a case where the information on the requested demands is unknown.

References

1. Tompkins, J.A., White, J.A., Bozer, et al. *Facilities Planning.* John Wiley & Sons: NJ. 2003.
2. Ten Hompel, M., Schmidt, T. *Warehouse Management- Automation and Organisation of Warehouse and Order Picking System*. Springer-Verlag: Berlin Heidelberg. 2007.
3. Petersen, C. G., Aase, G. "A comparison of picking, storage, and routing policies in manual order picking". *International Journal of Production Economics*, 92(1): p.11-19, 2004.
4. De Koster, R., Le-Duc, T. Roodbergen K.J. "Design and control of warehouse order picking". *European Journal of Operational Research*, 182(2): p.481-501, 2007.
5. Petersen, C.G. "An evaluation of order picking routing policies". *International Journal of Operations & Production Management*, 17 (11): p.1098–1111, 1997.
6. Goetschalckx, M., Ashayeri, J. "Classification and design of order picking systems". *Logistics World (June)*, p. 99–106, 1989.
7. De Koster, R. "How to assess a warehouse operation in a single tour". *Report*, RSM Erasmus University, the Netherlands.2004.
8. Gu, J., Goetschalckx, M., & McGinnis, L.F. "Research on warehouse operation: A comprehensive review". *European Journal of Operational Research*, 177: p.1–21, 2007.
9. Gademann, N., Van de Velde, S. "Batching to minimize total travel time in a parallel-aisle warehouse". *IIE Transactions*,37 (1): p.63–75, 2005.
10. Chen, M.C., Wu, H.P. "An association-based clustering approach to order batching considering customer demand patterns". *Omega International Journal of Management Science*, 33(4): p.333-343, 2005.
11. Le-Duc, T., De Koster, R. "An approximation for determining the optimal picking batch size for order picker in single aisle warehouses". In R., Meller,M.K.,Ogle, B.A.,Peters, G.D., Taylor, J., Usher(Eds.), *Progress in Material Handling Research*, pp. 267–286, 2003.
12. Parikh, Pratik J. "Designing order picking systems for distribution centers", *Ph.D. thesis*, Department of Industrial and Systems Engineering Virginia Tech, 2006.
13. Petersen, C.G. "Considerations in order picking zone configuration". *International Journal of Operations & Production Management*, 27 (7): p.793–805, 2002.
14. Jane, C.C. "Storage location assignment in a distribution center". *International Journal of Physical and Logistics Management*, 30 (1): p.55–71, 2000.

15. Jewkes, E., Lee, C., Vickson, "Product location, allocation and server home base location for an order picking line with multiple servers". *Computers & Operations Research*, 31: p.623–626. 2004.
16. Roodbergen, K.J. "Layout and routing methods for warehouses". *Ph.D. thesis*, RSM Erasmus University, the Netherlands, 2001.
17. Roodbergen, K.J., De Koster, R. "Routing methods for warehouses with multiple cross aisles". *International Journal of Production Research*, 39 (9): p.1865–1883, 2001.
18. Zhang, M., Batta, R., Nagi, R. "Modeling of Workflow Congestion and Optimization of Flow Routing in a Manufacturing/Warehouse Facility". *Management Science*, 55(2): p.267–280, 2009.
19. Smith, J., Li, W. "Quadratic assignment problems and M/G/C/C state dependent network flow". *J. Combined Optimization*, 5(4): p.421–443, 2001.
20. Chiang, W., Kouvelis, P., Urban, T. "Incorporating workflow interference in facility layout design: The quartic assignment problem". *Management Science*, 48(4): p.584–590, 2002.
21. Chiang, W., Kouvelis, P., Urban, T. "Single and multi-objective facility layout with workflow interference considerations". *Eur.J. Oper. Res.*, 174(3): p.1414–1426, 2006.
22. Herrmann, J., Ioannou, G., Minis, I., Nagi, R., et al. "Design of material flow networks in manufacturing facilities". *J. Manufacturing Systems*, 14(4): p.277–288, 1995.
23. Bakkalbasi, O. "Flow path network and layout configuration for material delivery systems", *Ph.D. thesis*, Georgia Institute of Technology, Atlanta, 1990
24. Vosniakos, G., Davies, B. "On the path layout and operation of an AGV system serving an FMS". *Internat. J. Advanced ManufacturingTech*, 4: pp. 243–262, 1989.
25. Kim, C., Tanchoco, J. "Operational control of a bi-directional automated guided vehicle system". *Internat. J. Production Res*, 31(9): p.2123–2138, 1993.
26. Beamon, B. "System reliability and congestion in a material handling system". *Comput. Indust. Engrg*, 36(3): p.673-684, 1999.
27. Pan, J.C-H., Wu, M-H. "Throughput analysis for order picking system with multiple pickers and aisle congestion considerations". *Computers & Operations Research*, 39: p.1661-1672, 2012.
28. Hong, S., Johnson, A.L., Peters, B.A., "Batch picking in narrow-aisle order picking systems with consideration for picker blocking". *European Journal of Operational Research*, 221(4): p.557-570, 2012.
29. Kim, B., Lee, W. "A multi-product dynamic inbound ordering and shipment scheduling problem at a third-party warehouse". *International Journal of Industrial Engineering: Theory, Applications and Practice*, 20(1-2), 2013.
30. Gokhan Ozden, S. "A computational system to solve the warehouse aisle design problem", *Ph.D. thesis*, Auburn University, 2017.
31. Scholz, A., Wascher, G. "Order batching and picker routing in manual order picking systems: the benefits of integrated routing". *Central European Journal of Operations Research*, 25: p.491-520, 2017.
32. Chen, F., Xu, G., Wei, Y. "An integrated metaheuristic routing method for multiple-block warehouse with ultranarrow aisles and access restriction", *Complexity*, ID 1280285, 2019.
33. Hojaghania, L., Nematian, J., Shojaiea, A., et al. "Metaheuristics for a new MINLP model with reduced response time for on-line order batching". *Scientia Iranica*, Online from December 2019.
34. Tajima, E., Suzuki, M., Ishighki, A., et al. "Effect of picker congestion on travel time in an order picking operation". *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, 14(5), 2020.
35. Zuniga, J., Martinez, J., Fierro, T., et al. "Optimization of the storage location assignment and the picker-routing problem by using mathematical programming". *Appl. Sci.*, 10(2), 2020.
36. Cano, J., Correa-Espinal, A., Gomez-Montoya, R. "Mathematical programming modeling for joint order batching, sequencing and picker routing problems in manual order picking systems", *Journal of King Saud University-Engineering Sciences*, 32(3): p.219-228, 2020.
37. Pai, A. S. "Development of deterministic collision-avoidance algorithms for routing automated guided vehicles". *MS.C. thesis*, Department of Industrial and Systems Engineering, Kate Gleason College of Engineering, 2008.
38. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., et al. "Equation of State Calculation by Fast Computing Machines". *J. of Chem. Phys.*, 21: p.1087-1092, 1953.
39. Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P. *Optimization by simulated annealing. Science*, 220: p.671-680, 1983.
40. Cerny, V. "A thermo dynamical approach to the traveling salesman problem: An efficient simulation algorithm", *Journal of Optimization Theory and Applications*, 4: p.41-45, 1985.
41. Aarts, E., Lenstra, J.K. *Local Search in Combinatorial Optimization*. New Jersey: Princeton University Press,2003.
42. Modiri-Delshad, M., Aghay Kaboli, S.Hr., Taslimi, E., et al. "An iterated-based optimization method for economic dispatch in power system". *IEEE Conference on Clean Energy and Technology (CEAT)*, 2013.
43. Modiri-Delshad, M., Aghay Kaboli, S.Hr., Taslimi-Renani, E., et al. "Backtracking search algorithm for solving economic dispatch problems with valve-point effects and multiple fuel options". *Energy*, 2016,116: p.637-649, 2016.
44. Pourdaryaei, A., Mokhlis, H., Azil Illias, H., et al. "Hybrid ANN and Artificial Cooperative Search Algorithm to Forecast Short-Term Electricity Price in De-Regulated Electricity Market". *IEEE Access* 7: 125369-125386, 2019.

45. Hlal, M. I., Ramachanaramurthya, V., Padmanaban, et al. "NSGA-II and MOPSO based optimization for sizing of hybrid PV/wind/battery energy storage system". *Int. J. Power Electron and Drive Syst*, 10.1: p.463-478, 2019.

Alireza Eydi is an Associate Professor of Industrial Engineering in the Faculty of Engineering at University of Kurdistan, Iran. He received his PhD from the Department of Industrial Engineering at Tarbiat Modares University, Iran in 2009. His main areas of teaching and research interests include supply chain and transportation planning and network optimization problems including routing and location problems on networks.

Hanif Mohagheghi received his MSC degree of Industrial Engineering from University of Kurdistan, Iran.

Seyed Ali Ghasemi-Nezhad received his MSC degree of Industrial Engineering from University of Kurdistan, Iran. His main areas of expertise include transportation planning, strategic planning, feasibility study of projects, and portfolio management.

Figure1. Representation of sample problem

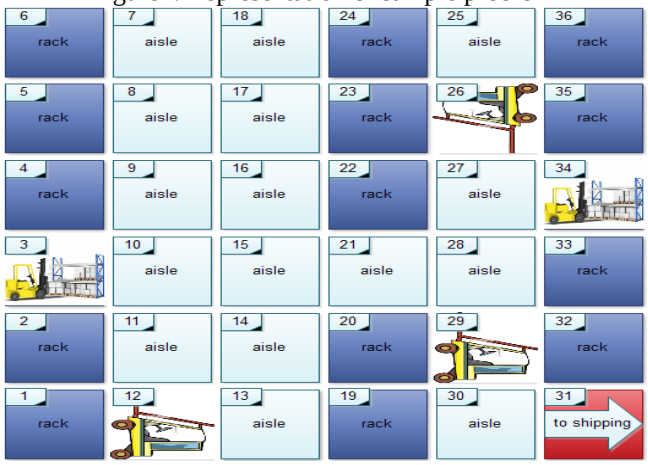

Table 1. Model results for sample problems (exact approach)

Problem characteristics									Exact results	
No.	Warehouse dimension	Grid size	Number of vehicles	Number of items to be picked	Number of rack columns	Number of narrow aisles	Number of cross aisles	Grid number of the items in the pick list	Objective value	Solution time (second)
1	3*3	9	1	1	1	0	1	1	7	0.106
2	3*3	9	1	2	1	0	1	1-4	12	0.236
3	3*3	9	1	3	1	0	1	1-4-7	15	0.375
4	5*4	20	1	3	3	2	1	3-10-11	26	6.57
5	5*4	20	2	3	3	2	1	3-10-11	15	13.49
6	5*4	20	3	3	3	2	1	3-10-11	13	209.1
7	6*4	24	1	3	3	1	1	4-12-13	31	19.27
8	6*4	24	2	3	3	1	1	4-12-13	15	40.96
9	6*4	24	3	3	1	1	1	4-12-13	13	213.64
10	6*6	36	1	3	3	3	2	11-15-19	39	186.37
11	6*6	36	2	3	3	3	2	11-15-19	20	124.51

12	6*6	36	2	3	3	3	2	11-15-19	22	1178.38
13	6*6	36	3	3	3	3	2	11-15-19	15	42.21
14	6*6	36	4	2	3	3	2	11-15	15	39.13
15	8*6	48	1	1	4	4	2	9	21	0.903
16	8*6	48	1	2	4	4	2	9-21	36	11.79
17	8*6	48	1	3	4	4	2	21-31-35	27	49.75
18	8*6	48	1	3	4	4	2	9-21-31	37	228.447
19	8*6	48	1	3	4	4	2	9-21-35	43	583.21
20	8*6	48	1	4	4	4	2	9-21-31-35	47	17124

Figure 2. Representation of warehouse configuration for problems 7, 8 and 9

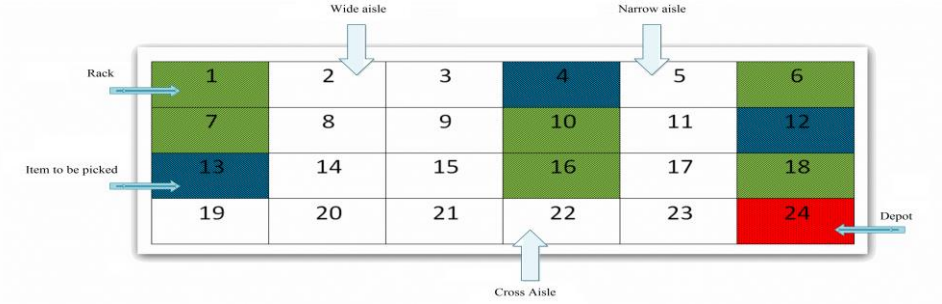

Figure 3. Representation of initial solution of problem #8

24	24	24	24	23	22	23	22	21	15	14	14	15	21	22	23	24	...	...	22
23	17	11	11	17	23	24	233	17	11	11	11	11	11	11	11	11	...	...	24
1001	14	1002	11	5	0	0	0	0	0	0	0	0	0	0	0	0	...	...	0

Number of cells

Figure 4. Representation of warehouse grid layout (set S_i) of problem #8 in C++ code

0	0	0	0	0	0
2	3	8	0	0	3
2	3	9	0	0	3
5	0	0	0	0	1
5	11	0	0	0	2
0	0	0	0	0	0
0	0	0	0	0	0
2	8	9	14	0	4
3	8	9	15	0	4
0	0	0	0	0	0
5	11	17	0	0	3
11	0	0	0	0	1
14	0	0	0	0	1
8	14	15	20	0	4
9	14	15	21	0	4
0	0	0	0	0	0

11	17	23	0	0	3
0	0	0	0	0	0
19	20	0	0	0	2
14	19	20	21	0	4
15	20	21	22	0	4
21	22	23	0	0	3
17	22	23	24	0	4
23	24	0	0	0	2

Figure 5. Neighborhood structure generation of problem #8

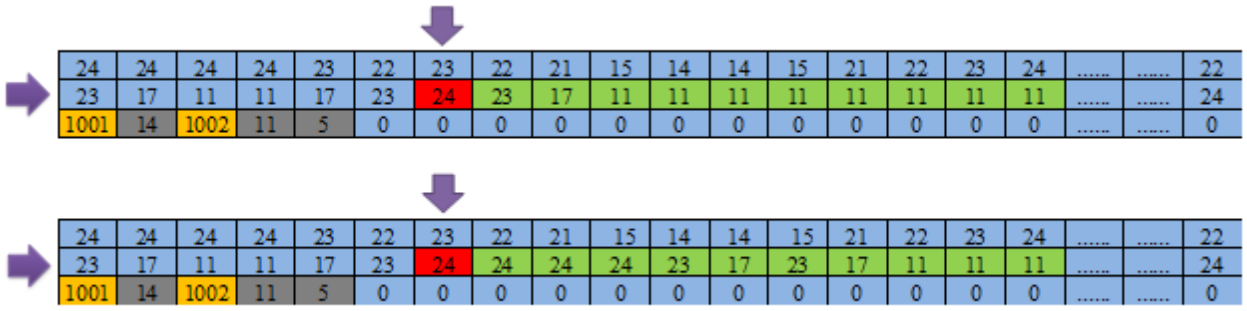

Figure 6. Swapping structure generation of problem #8

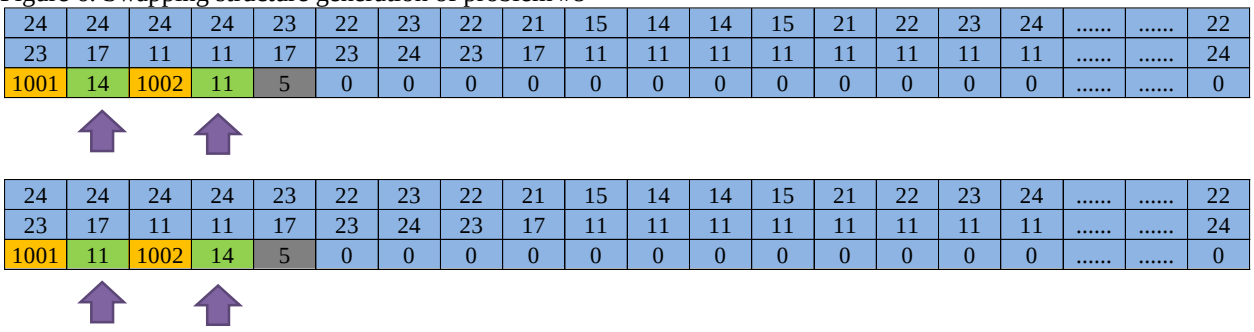

Table 2. Computational results of SA algorithm

Problem characteristics							**Objective function of SA algorithm**			
No.	Warehouse dimension	Grid size	Number of vehicles	Number of items to be picked	Grid number of the items in the pick list	Upper bound (Tmax)	Best Solution	Worst Solution	Average of the solutions	Solution time(second)
1	3*3	9	1	1	1	12	7	7	7	7

2	3*3	9	1	2	1-4	18	12	12	12	9
3	3*3	9	1	3	1-4-7	18	15	15	15	10
4	5*4	20	1	3	3-10-11	30	26	26	26	38
5	5*4	20	2	3	3-10-11	30	16	16	16	38
6	5*4	20	3	3	3-10-11	30	13	14	12.6	65
7	6*4	24	1	3	4-12-13	35	31	31	31	55
8	6*4	24	2	3	4-12-13	35	15	16	15.8	74
9	6*4	24	3	3	4-12-13	35	16	16	16	99
10	6*6	36	1	3	11-15-19	45	39	39	39	103
11	6*6	36	2	3	11-15-19	45	22	32	23.2	137
12	6*6	36	2	3	11-15-19	45	23	23	23	135
13	6*6	36	3	3	11-15-19	45	16	18	16.5	181
14	6*6	36	4	2	11-15	45	16	17	16.9	204
15	8*6	48	1	1	9	30	21	21	21	100
16	8*6	48	1	2	9-21	40	36	36	36	115
17	8*6	48	1	3	21-31-35	40	27	27	27	118
18	8*6	48	1	3	9-21-31	40	37	37	37	119
19	8*6	48	1	3	9-21-35	45	43	43	43	125
20	8*6	48	1	4	9-21-31-35	45	48	48	48	130

Table 3. Comparing results of exact algorithm and SA algorithm

Problem characteristics					**Exact results**		**SA Metaheuristic**		
No.	Warehouse dimension	Grid size	Number of vehicles	Number of items to be picked	Objective value	Solution time (second)	Objective Value	Solution time (second)	Optimality Gap%
1	3*3	9	1	1	7	0.106	7	7	0
2	3*3	9	1	2	12	0.236	12	9	0
3	3*3	9	1	3	15	0.375	15	10	0
4	5*4	20	1	3	26	6.57	26	38	0
5	5*4	20	2	3	15	13.49	16	38	6.67
6	5*4	20	3	3	13	209.1	13.6	65	4.61
7	6*4	24	1	3	31	19.27	31	55	0
8	6*4	24	2	3	15	40.96	15.8	74	5.34
9	6*4	24	3	3	13	213.64	16	99	23.07

10	6*6	36	1	3	39	186.37	39	103	0
11	6*6	36	2	3	20	124.51	23.2	137	16
12	6*6	36	2	3	22	1178.38	23	135	4.54
13	6*6	36	3	3	15	42.21	16.5	181	10
14	6*6	36	4	2	15	39.13	16.9	204	12.67
15	8*6	48	1	1	21	0.903	21	100	0
16	8*6	48	1	2	36	11.79	36	115	0
17	8*6	48	1	3	27	49.75	27	118	0
18	8*6	48	1	3	37	228.447	37	119	0
19	8*6	48	1	3	43	583.21	43	125	0
20	8*6	48	1	4	47	17124	48	130	2.12

Chart (1). Comparison of results between exact and SA algorithm

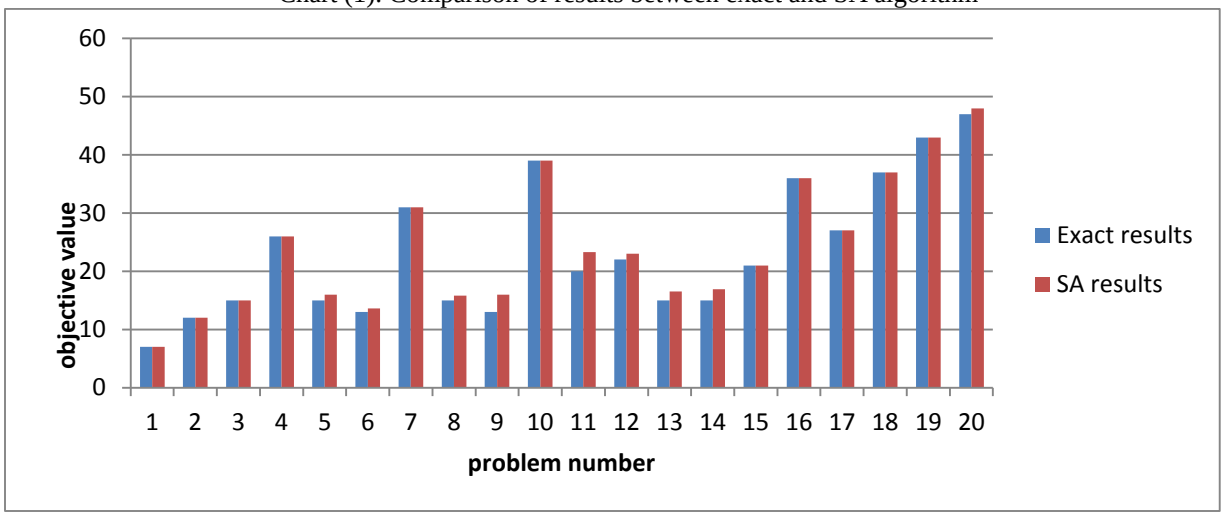

Chart (2). Comparing solution times between exact and SA algorithm