RESEARCH ARTICLE

Route planning methods for a modular warehouse system

Elif G. Dayıoğlu [a *], Kenan Karagül [b], Yusuf Şahin [c], Michael G. Kay [d]

[a] Department of Computer Technology and Information Systems, Mersin University, Turkey
[b] Department of Logistics, Pamukkale University, Turkey
[c] Department of Business Administration, Burdur Mehmet Akif Ersoy University, Turkey
[d] Edward P. Fitts Department of Industrial&Systems Engineering, NC State University, USA
edayioglu@mersin.edu.tr, kkaragul@pau.edu.tr, ysahin@mehmetakif.edu.tr, kay@ncsu.edu

ARTICLE INFO	ABSTRACT
	In this study, procedures are presented that can be used to determine the routes of the packages transported within a modular storage system. The problem is a variant of robot motion planning problem. The structures of the procedures are developed in three steps for the simultaneous movement of multiple unit-sized packages in a modular warehouse. The proposed heuristic methods consist of route planning, tagging, and main control components. In order to demonstrate the solution performance of the methods, various experiments were conducted with different data sets and the solution times and qualities of the proposed methods were compared with previous studies. It was found that the proposed methods provide better solutions when taking the number of steps and solution time into consideration.

1. Introduction

Logistics is the process of strategically managing the supply, transport, and storage of raw materials, semi or finished products to ensure cost-effectiveness. The raw materials and semi-finished products used by a company and the finished products produced by the company must be moved from one location to another. Logistic activities, which have a significant impact on the success of the production and distribution operations of the company, are composed of many functional areas. The performance shown in these functional areas leads to an increase in service quality as well as a reduction in operating costs, and logistics has to provide high-quality service at a low or acceptable cost [1].

In logistic operations, it is an important challenge to meet the different products demanded by consumers [29]. One of the most critical functions in logistics processes is warehousing. During this process, the products are stored at certain points for a certain period of time. The primary purpose of the classical warehousing is to store the products in a correct and non-destructive way. On the other hand, many operations are carried out from the receipt of the products to the delivery of them to the customer in today's modern warehousing concept. Such a system requires a high level of coordination between the seller and the buyer's decision-making [30]. In modern warehouse systems, activities such as classification of products, quality control, packaging, barcoding, labeling, keeping records of stock movements, providing the communication with the related units (sender, buyer, customer, producer, etc.) are carried out in addition to other activities [2,3].

It is possible to classify warehouses according to geographical distribution (central and decentralized), property structure (unique, general, and contract), product characteristics (parts, bulk) and operation (production, distribution). A public logistics networks (PLN) is a network that provides an alternative to private logistics networks for the transport of goods. A PLN consists of distribution centers (DCs), trucks, and package components. In these networks, which are inspired by the structure and operation of the Internet, a package is sent from a store to a public distribution center located in an area in a metropolitan area [4].

The use of automation systems for the activities carried out at the distribution centers provides a significant reduction in costs [5,6]. Fully automated warehouses (loading, unloading, sorting, stacking, automation of

*Corresponding author

packages storage and retrieval) have become an essential issue for effective cost minimization and warehouse management. These warehouses where the operations in the warehouse are fully automated are defined as modular warehouses. Kay [5] suggested a distribution center design that would meet these requirements. The proposed system consists of square modules with orthogonal pop-up powered wheels. Figure 1 shows the top view of one of the modules with orthogonal pop-up powered wheels. In each direction, the wheels of the module are raised and lowered relative to the wheels in the other direction. The guides (Fig. 2) used in this system can be raised and lowered when necessary to limit and direct the movement of the load [7].
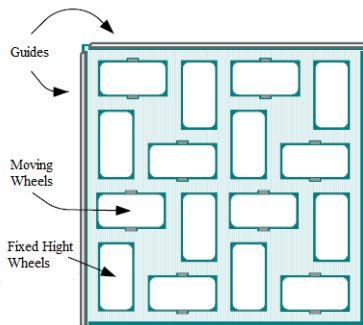

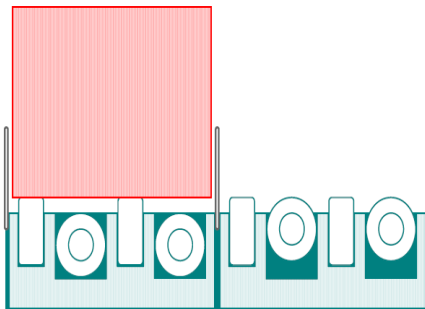**Figure 1.** Top view of a single module [5]

Figure 2. Guidelines in the ascended state [5]

In this study, heuristics algorithms based on the algorithm proposed by Datar [6] and Sittivijan [8] are used for the control of packages in a modular warehouse. The purpose of the problem addressed is that the packages can be delivered to the desired exit point in the shortest time and least number of steps. The conceptual framework of the study is presented in Section 2. In this context, 15-floating block, rush hour, and the warehouseman's problem along with studies related to these problems are presented. The details of the proposed methods are given in Section 3. Experimental studies and results were included in the fourth and fifth sections, respectively.

2. Route planning problem

In this study, heuristic algorithms based on the transport of unit-size packages in a modular warehouse with a limited number of free spaces are proposed. Therefore, the problem is closely related to the motion planning problem. In this section, 15-floating block,

rush hour and warehouseman's problems and the related literature are examined. These problems will provide a better understanding of this problem, which is considered within the scope of the study and related to the movement of objects within a limited space.

2.1. 15-floating block problem

The 15-floating block problem is similar to the structure of the problem discussed in this study. It is a purer form of the problem of transporting more than one object in a limited area [6]. In this problem, a square area of 4×4 has 15 full tiles and one empty tile numbered from 1 to 15, which will be rearranged according to a specific target configuration. An adjacent tile can be shifted to this position orthogonally by the described empty tile [9]. The goal is to reach the final target by moving the tiles only horizontally or vertically from the initial state as shown in Figure 3.
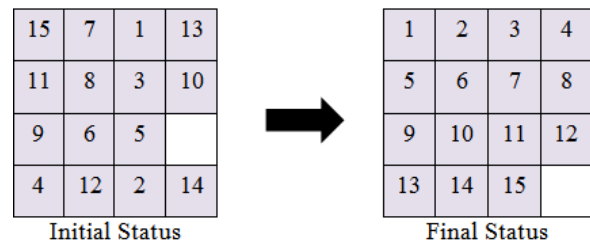

Figure 3. Floating block puzzle

There are many studies using various methods in the literature regarding this problem. Spitznagel [10] has proved that it is only possible to obtain the end configuration from the initial configuration by double-numbered permutation. Reinefeld [11] discussed the 8-floating block problem and evaluated the utility of node sequencing using the recursive deepening A* (YDA*) algorithm. It has been concluded that YDA* applications performed with a fixed operator (e.g., up, left, right, down) perform worse than those done with a simple random operator selection. Gue and Kim [12] developed a 15-block based warehouse system. Unlike the floating block problem, the calculation is made for more than one free space and as the number of free space increases, the retrieval time is reduced. Bauer [9] proposed the Manhattan Pair Distance Heuristics (MCU), which is a combination of YDA algorithm and Manhattan distance function. With the help of the proposed method, the number of nodes in the heuristic search has been reduced by 80% for the 15-floating block problem.

2.2. Rush hour problem

The rush hour problem, as seen in Figure 4, is a module-based game that consists of a target vehicle to be transported to the exit point and only a few vehicles moving in the horizontal or vertical direction [13]. Other cars in the module are moved to open the path to the designated exit of the target car.
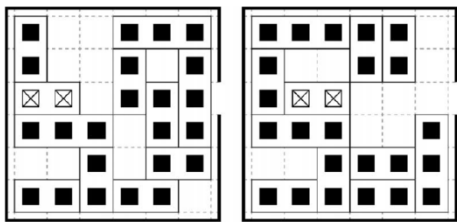
Figure 4. Rush hour problem

Flake and Baum [13] showed that the decision of whether the target vehicle would exit the module was PSPACE-Complete. Furthermore, unlike the original rush hour problem, they presented a generalized version of the traffic problem (GSH - Generalized Rush Hour Problem) with the option of arbitrary width and height and the possibility of the outlet to be anywhere in the vicinity of the grid. Hearn and Demaine [14] proposed a nondeterministic calculation model based on the inverse edge directions in the weighted directional charts with minimum flow constraints. The framework they developed was inspired by "Generalized Rush Hour Logic" developed by Flake and Baum [13]. Hauptman et al. [15] designed a novel IDA*-based heuristics solver for the Rush Hour domain.

2.3. Warehouseman's problem

The warehouseman's problem, which is an extension of the $n \times n$ floating block problem, involves coordinated movement planning of a large number of independent objects in a limited area [16]. The goal is to move objects in the repository from the initial configuration to the final configuration [6]. Coordinated motion planning of a large number of three-dimensional objects in the presence of obstacles is a computational problem in which it is important to regulate complexity [17]. Hopcroft et al. [17] proved that the problem of simultaneous motion planning for a limited number of discrete rectangular bodies of different sizes to move within a 2-dimensional rectangular area is PSPACE-hard. Yeung and Bekey [18] used a decentralized approach based on the problem being global and local road planning problem. Sanchez and Latombe [19] used probabilistic roadmaps (PRM) which plans free paths for multiple interacting robots without collision. They developed a new PRM planner that combines a single-query bi-directional sampling strategy with a lazy collision-checking connection strategy. Sharma and Aloimonos [20] proposed a solution method introducing constraints on the size of objects for non-unit sized objects and distributing the free space for warehouseman problem. Sarrafzadeh and Maddila [21] formed a two-dimensional warehouse system consisting of square objects (robots and obstacles) that were allowed to move horizontally and vertically along the grid lines.

LaValle and Hutchinson [22] used a dynamic programming-based solution algorithm to solve multiple robot motion planning problems. Azarm and Schmidt [23] developed a framework that is decentralized and allows for parallel decision for multiple robots to solve the collision problem. The framework allows parallel path computation and dynamic priority assignment. Svestka and Overmars [24] proposed a coordinated approach to the problem of multi-robot road planning, unlike conventional decentralized planning. In the proposed system, the data structure that stores multi-robot motion information is created in two steps. In the first step, a roadmap for only one robot is created using the PRM planner, and then some of these simple roadmaps have been made a roadmap for the composite robot in the second step.

Leroy et al. [25] developed a geometric-based method for motion planning of multiple robots. While the paths of all the robots are calculated independently of each other, the problem of coordinating the movements of the robots in their way so as not to collide with each other has been emphasized. Guo and Parker [26] proposed a distributed and best motion planning algorithm for multiple robots. This computational complexity problem is divided into two modules as path and speed planning, and D* search method is applied to both modules. Yamashita et al. [27] suggested a two-stage method for motion planning of multiple mobile robots in order to move a large object together in a three-dimensional environment. As a result, they have integrated their movement planner into two phases as a global road planner and a local motion planner. In global path planning, constraints of object motion are considered as a cost function and a heuristic function in the A* search. Liu et al. [28] presented a road planning scheme based on the ant colony algorithm with collision avoidance for multiple robot systems. In order to solve the collision between moving robots, they adopted a behavior strategy on "first come and first served" principle.

3. Proposed methods

In this study, five solution methods based on A* heuristic are proposed for planning the movement of packets to avoid collisions and deadlocks in a modular storage system. Although the proposed methods are diversified in some respects, they have the same components. With the help of these approaches, motion plans are prepared in the warehouse system where there are moving obstacles consisting of more than one object moving at the same time. Since there may be conflicts between moving objects, the route planning is not sufficient to bring the active objects to the desired targets at this stage. Therefore, the methods include components such as route planning, tagging, and main control. These components are described in the following sections.

3.1. Route planning phase

Each of the active objects has a planned path. An active object has the capability of planning a route from its initial position to its destination. On the other hand, if it has been tagged by a higher priority object and move

away from its current planned path, it can plan a new route from its initial location or its current location. Route planning is used to find this path. To find the path from the current position to the target position, the orthogonal neighboring modules around the current module are examined. In this study, A* based heuristic algorithms are used to select the next module to be moved. The lowest cost neighbor module is selected as the new module with this algorithm. The standard A* algorithm was modeled by making some arrangements because the warehouse system discussed in the study was not static. The location of the objects in the warehouse changes with time, so it is not a static but a dynamic environment. Therefore, a new function named $LB_{(x,y)}^T$ is used instead of the $F(n)$ function. The module with the smallest $LB_{(x,y)}^T$ value is selected as the module to be moved. According to Equation (1), the current position (a, b) of the object to be moved and the target position (x, y) of the object is assumed to be as follows:

$$LB_{(x,y)}^T = T_{(a,b)}^{(a_0,b_0)} + T_{(x,y)}^{(a,b)} + T_{(a_n,b_n)}^{(x,y)} \quad (1)$$

where

$T_{(a,b)}^{(a_0,b_0)}$: the wandering time from the starting module (a_0, b_0) to some intermediary position (a, b)

$T_{(x,y)}^{(a,b)}$: the weighted estimated wander time to go to the neighboring module (x, y) during the next k time steps.

Since the configuration of the objects in the system may vary from one time step to another time step, at this stage, the weighted sums for each t time step are computed using Equation (2).

$$T_{(x,y)}^{(a,b)} = \sum_{t=1}^{k} w_t \times T_{(x,y),t}^{(a,b)} \quad (2)$$

While the $T_{(x,y)}^{(a,b)}$ value is calculated, the occupancy gap state of the neighboring module is considered during the time off from the current time step ($t = 1$) to the kth time step ($t = k$). Because the state of the objects in the system can vary greatly from one time step to another time step. In this paper, the route planning is taken as $k = 3$ and the system state in each of the 3-time steps from the time step present in the route planning for each package is evaluated. The w_t value in Equation (2) is arbitrarily chosen, but it must satisfy the conditions of $\sum_{t=1}^{k} w_t = 1$ and $w_t > w_{t-1}$.

$T_{(x,y),t}^{(a,b)}$: estimated wander time to move the object from its current module (a, b) to the neighbor module (x,y) at the time step t.

$T_{(x,y),t}^{(a,b)} = 1$: At $t = k$ time step if the neighboring module is empty, the current module passes in a time step with the neighboring module.

$T_{(x,y),t}^{(a,b)} \geq 2$: At t = k time step, if the neighboring module is filled with a low-priority object, switching to that module takes

place in one or more time steps.

$T_{(x,y),t}^{(a,b)} = \infty$: At $t = k$ time step, if there is a high priority object that has reached the target in the neighboring module, it can not be moved, and this variable takes the infinite value.

$T_{(a_n,b_n)}^{(x,y)}$: The distance from the neighboring module to the target point. The Manhattan distance method is used to calculate by Equation (3).

$$T_{(a_n,b_n)}^{(x,y)} = |x_{neighbour} - x_{target}| \\ + |y_{neighbour} - y_{target}| \quad (3)$$

$LB_{(x,y)}^T$ is the lower bound value used on the path to be defined to go from point (a, b) to point (x, y). As in the case where the neighbor with the smallest $F(n)$ value is selected in the A* algorithm, here the neighbors with the smallest $LB_{(x,y)}^T$ value from the orthogonal neighbors of the current module is selected too.

3.2. Tagging phase

When moving on the planned path of the high priority object, if it encounters a lower priority or inactive object from the active object on the path, this process is used to move these objects away from the defined path. In Figure 5, the object numbered 8 tries to move from module (2,2) to module (2,3). Module (2,3) has an inactive object with 4 priority. For this reason, the object with 8 priority tags the object with 4 priority. In tagging, 8, which is the priority of the current object, is transferred to the object with 4 priority as the inheritance priority. Thus, the object with 4 priority can move 5, 6 or 7 priority objects. Because this object has a value of 8 as the inheritance priority during the tagging process. After the object with a priority of 4 has been tagged, it is checked whether they are empty neighbors that can move. Neighbors are (3,3) and (1,3) modules. The object with priority 4 selects the object with priority 2, which is the lowest priority neighbor. 4's inheritance priority passes to object with priority 2, but when the object with priority 2 tries to tag the object with priority 9, returns to the object with priority 4 because 9's priority is higher than 8. Here, backtracking is performed. The new object to be tagged is selected as 7, 7 → 5, 5 → 3, 3 → 6 tags and the module (3,1) is found as the last empty module. The tagging process ends in this way. In the latest case, it is moved to 6 → (3,1), 3 → (2,1).
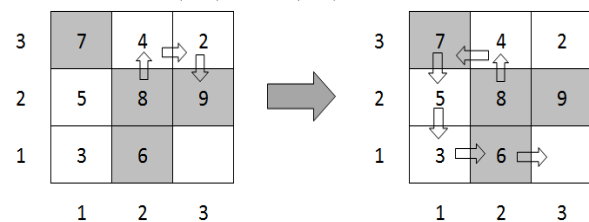

Figure 5. Example of the tagging process

3.3. Main control phase

The movement of all active and inactive objects is controlled by the main control component. At each time step, this part controls every active object that is not at the destination point and checks whether it is tagged by another high priority object. If it is not tagged, it performs route planning for the currently active object due to the changing system environment.

Possible situations in the main control process can be listed as follows:

✓ If the neighboring module is empty and not tagged by another object, then the currently active object passes to the neighboring module within that time step.

✓ If the neighboring module is empty and tagged by another object, then it is checked whether the neighboring module is tagged by the low priority. If the neighboring module is tagged by the low priority object, the tagging process of that object is released, and the currently active object is passed to the neighboring module at that time step. If the neighboring module is tagged by a high priority object, the higher priority object is expected to move from that module.

✓ If the neighboring module is not empty and is tagged by another object, it is also checked whether the neighboring module is tagged by the low priority. If the neighboring module is tagged by the low priority object, the tagging process for that object is released, and the tagging process is performed by the current object, and the current module is moved to the neighbor module. If the neighboring module is tagged by a high priority object, the higher priority object is expected to move from that module.

✓ If the neighboring module is not empty and is not tagged by another object, then the object's priority in the neighboring module is looked. If the priority of the neighboring module is lower than the priority of the active object, the labeling process is started by the active object for this module and if the tagging process is successful, the active object passes to the neighboring module. If the priority of the neighboring module is higher than the priority of the active object, the higher priority object is expected to move from that module.

4. Experimental study

In order to show the performance of the proposed algorithms, 23 test problems were produced for 3 group dataset. The dataset is divided into groups according to the density and dimensions of the warehouse. Table 1 shows the group numbers of the dataset and the size and

density information of the warehouses. The first group contains 40% and 50% density warehouse test problems in 4×4 sizes. The second group has a 6×6 sizes of warehouse layout with the density ranging from 40-70%. Moreover, the last one consists of 20-99% density and 16×32 warehouse sizes.

Table 1. Details of the data set

The group of data sets	Number of data set	Density Interval (%)	Size of warehouse
Group 1	1, 2 and 3	40-50	4×4
Group 2	4, ..., 8	40-70	6×6
Group 3	9, ..., 23	20-99	16×32

All algorithms were implemented in the Eclipse environment using the Java programming language. Comparisons were made on a standard computer with 4 GB RAM and 2.67 GHz processor. In Table 2, the features and differences of all examined algorithms are shown in summary.

Table 2. Details of algorithms

Algorithm	Features and Differences
ALG-B1 (1)	✓ Sittivijan (2015) algorithm ✓ For each active object, an A * based intuitive route planning is performed at the beginning
ALG-B2 (2)	✓ Datar (2011) algorithm ✓ It is a greedy approach. ✓ It is an algorithm that is planned only for the movement at the next time step.
ALG-P1 (3)	✓ The algorithm in which ALG-B1 is restarted in every environment change ✓ For each active object, the route planning is performed again with an intuitive A * based always on the time step
ALG-P2 (4)	✓ An improved version of ALG-B1. ✓ Release process applied to tag object is removed from the main control and applied only during the tagging process
ALG-P3 (5)	✓ An improved version of ALG-P1 ✓ Release process applied to tag object is removed from the main control and applied only during the tagging process
ALG-P4 (6)	✓ An improved version of ALG-B1 ✓ The calculation of the $LB_{(x,y)}^{T}$ function has been changed
ALG-P5 (7)	✓ An improved version of ALG-P3 ✓ The calculation of the $LB_{(x,y)}^{T}$ function has been changed

In the experimental study, solution times and the number of steps in reaching the final solution were taken into consideration for the performance comparison of the methods. In Table 3, all algorithms were compared for data sets in terms of the number of steps required to reach the destination points of the packages.

Table 3. Number of solution steps

Data Set	METHODS							Size	Density
	1	2	3	4	5	6	7		
1	8	9	8	8	8	8	8	4*4	0,4
2	8	9	7	8	7	7	8	4*4	0,5
3	9	4	9	9	9	3	3	4*4	0,5
4	11	20	11	11	11	11	11	6*6	0,4
5	12	13	12	12	12	12	12	6*6	0,5
6	6	14	6	6	6	6	6	6*6	0,6
7	19	16	15	19	15	19	15	6*6	0,6
8	13	19	13	13	13	13	13	6*6	0,7
9	29	31	27	29	27	29	27	16*32	0,2
10	29	31	27	29	27	29	27	16*32	0,2
11	26	31	24	26	24	26	24	16*32	0,2
12	29	33	25	29	25	29	25	16*32	0,3
13	32	37	33	32	33	32	33	16*32	0,4
14	47	40	37	47	37	47	37	16*32	0,5
15	41	47	38	41	38	41	38	16*32	0,6
16	53	75	52	53	52	53	52	16*32	0,7
17	64	122	75	64	75	64	75	16*32	0,8
18	89	117	80	89	80	89	82	16*32	0,9
19	143	128	125	143	125	143	125	16*32	0,95
20	-	232	205	-	205	-	208	16*32	0,96
21	-	-	-	-	-	-	-	16*32	0,97
22	-	-	-	-	-	-	-	16*32	0,98
23	-	-	-	-	-	-	-	16*32	0,99
	582	692	543	582	543	582	545		

Table 4. Relative comparison of solution steps

Data Set	METHODS							Size	Density
	1	2	3	4	5	6	7		
1	1	1,13	1,00	1	1,00	1,00	1,00	4*4	0,40
2	1	1,13	0,88	1	0,88	0,88	1,00	4*4	0,50
3	1	0,44	1,00	1	1,00	0,33	0,33	4*4	0,50
4	1	1,82	1,00	1	1,00	1,00	1,00	6*6	0,40
5	1	1,08	1,00	1	1,00	1,00	1,00	6*6	0,50
6	1	2,33	1,00	1	1,00	1,00	1,00	6*6	0,60
7	1	0,84	0,79	1	0,79	1,00	0,79	6*6	0,60
8	1	1,46	1,00	1	1,00	1,00	1,00	6*6	0,70
9	1	1,07	0,93	1	0,93	1,00	0,93	16*32	0,20
10	1	1,07	0,93	1	0,93	1,00	0,93	16*32	0,20
11	1	1,19	0,92	1	0,92	1,00	0,92	16*32	0,20
12	1	1,14	0,86	1	0,86	1,00	0,86	16*32	0,30
13	1	1,16	1,03	1	1,03	1,00	1,03	16*32	0,40
14	1	0,85	0,79	1	0,79	1,00	0,79	16*32	0,50
15	1	1,15	0,93	1	0,93	1,00	0,93	16*32	0,60
16	1	1,42	0,98	1	0,98	1,00	0,98	16*32	0,70
17	1	1,91	1,17	1	1,17	1,00	1,17	16*32	0,80
18	1	1,31	0,90	1	0,90	1,00	0,92	16*32	0,90
19	1	0,90	0,87	1	0,87	1,00	0,87	16*32	0,95
20	-	-	-	-	-	-	-	16*32	0,96
21	-	-	-	-	-	-	-	16*32	0,97
22	-	-	-	-	-	-	-	16*32	0,98
23	-	-	-	-	-	-	-	16*32	0,99
Mean	1,00	1,23	0,95	1,00	0,95	0,96	0,92		

All algorithms were compared with ALG-B1 (1) according to the solution step numbers, and the results are shown in Table 4 and Figure 6. When the solutions are examined in terms of the number of steps, it has been observed that the proposed algorithms generally have better results than ALG-B1 (1) and ALG-B2 (2). For example, while the proposed methods reached a solution in 15 steps for the 7th dataset, ALG-B1 (1) and ALG-B2 (2) were able to reach solutions in steps of 19 and 16, respectively. For some datasets, the solution could not be obtained. The reason for this is that in the present configuration, no path can be defined for the arrival of the active packets to the destination points. A deadlock event occurs for these datasets. As a result, packages cannot move to any module.

Figure 6. Relative comparison of the solution steps

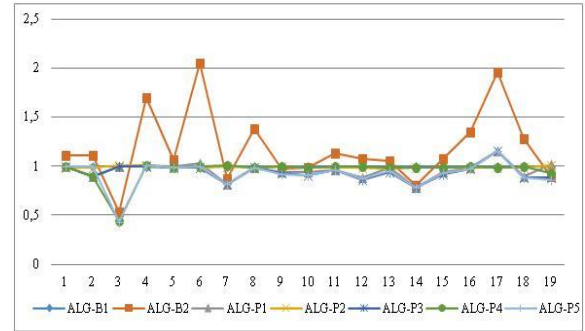The second comparison of the obtained solutions is the solution times. The solution times of the methods for different datasets are shown in Table 5. On the other hand, the relative comparison is made according to the ALG-B1 method in Table 6. When the average solution times are taken into consideration, it is seen that the proposed methods provide better solutions in shorter times. The solution time comparison is shown in Figure 7.

Table 5. Solution times of the methods

Data Set	METHODS							Size	Density
	1	2	3	4	5	6	7		
1	4,88	5,40	4,89	4,86	4,88	4,90	4,87	4*4	0,4
2	4,91	5,45	4,41	4,88	4,39	4,38	4,92	4*4	0,5
3	5,40	2,84	5,41	5,43	5,38	2,38	2,32	4*4	0,5
Total	**15,1**	**13,7**	**14,7**	**15,1**	**14,6**	**11,6**	**12,1**		
4	1	1,82	1,00	1	1,00	1,00	1,00	6*6	0,4
5	1	1,08	1,00	1	1,00	1,00	1,00	6*6	0,5
6	1	2,33	1,00	1	1,00	1,00	1,00	6*6	0,6
7	1	0,84	0,79	1	0,79	1,00	0,79	6*6	0,6
8	1	1,46	1,00	1	1,00	1,00	1,00	6*6	0,7
Total	**36,2**	**46,9**	**34,2**	**35,9**	**33,9**	**36**	**34,01**		
9	1	1,07	0,93	1	0,93	1,00	0,93	16*32	0,2
10	1	1,07	0,93	1	0,93	1,00	0,93	16*32	0,2
11	1	1,19	0,92	1	0,92	1,00	0,92	16*32	0,2
12	1	1,14	0,86	1	0,86	1,00	0,86	16*32	0,3
13	1	1,16	1,03	1	1,03	1,00	1,03	16*32	0,4
14	1	0,85	0,79	1	0,79	1,00	0,79	16*32	0,5
15	1	1,15	0,93	1	0,93	1,00	0,93	16*32	0,6
16	1	1,42	0,98	1	0,98	1,00	0,98	16*32	0,7
17	1	1,91	1,17	1	1,17	1,00	1,17	16*32	0,8
18	1	1,31	0,90	1	0,90	1,00	0,92	16*32	0,9
19	1	0,90	0,87	1	0,87	1,00	0,87	16*32	0,95
20	-	-	-	-	-	-	-	16*32	0,96
21	-	-	-	-	-	-	-	16*32	0,97
22	-	-	-	-	-	-	-	16*32	0,98
23	-	-	-	-	-	-	-	16*32	0,99
Total	**344,8**	**673,8**	**456,1**	**342,3**	**439,1**	**336,5**	**437,1**		

Table 6. Relative comparison of solution times

Data Set	METHODS							Size	Density
	1	2	3	4	5	6	7		
1	1,00	1,11	1,00	0,99	1,00	1,00	1,00	4*4	0,4
2	1,00	1,11	0,90	0,99	0,89	0,89	1,00	4*4	0,5
3	1,00	0,53	1,00	1,01	1,00	0,44	0,43	4*4	0,5
4	1,00	1,70	1,01	1,01	1,00	1,01	1,01	6*6	0,4
5	1,00	1,06	1,00	0,99	0,99	0,99	0,99	6*6	0,5
6	1,00	2,05	1,03	0,99	0,99	1,00	1,00	6*6	0,6
7	1,00	0,87	0,81	1,00	0,81	1,01	0,82	6*6	0,6
8	1,00	1,38	0,99	0,98	0,99	0,98	0,98	6*6	0,7
9	1,00	0,97	0,94	1,00	0,93	1,00	0,92	16*32	0,2
10	1,00	0,98	0,94	0,98	0,91	0,98	0,91	16*32	0,2
11	1,00	1,13	0,96	0,99	0,96	1,00	0,96	16*32	0,2
12	1,00	1,08	0,88	0,98	0,86	1,00	0,87	16*32	0,3
13	1,00	1,05	0,98	0,97	0,94	0,99	0,95	16*32	0,4
14	1,00	0,80	0,78	0,99	0,78	0,98	0,78	16*32	0,5
15	1,00	1,08	0,94	0,98	0,92	0,98	0,93	16*32	0,6
16	1,00	1,35	0,98	0,99	0,97	1,00	0,97	16*32	0,7
17	1,00	1,96	1,16	0,99	1,16	0,99	1,16	16*32	0,8
18	1,00	1,28	0,89	0,99	0,88	1,00	0,88	16*32	0,9
19	1,00	0,90	1,02	1,01	0,88	0,93	0,86	16*32	0,95
20	-	-	-	-	-	-	-	16*32	0,96
21	-	-	-	-	-	-	-	16*32	0,97
22	-	-	-	-	-	-	-	16*32	0,98
23	-	-	-	-	-	-	-	16*32	0,99
Total	**344,8**	**673,8**	**456,1**	**342,3**	**439,1**	**336,5**	**437,1**		

Figure 7. Relative comparison of the solution times

5. Conclusion

In this paper, the modular warehouse management issue is discussed, and the new A* based heuristics algorithms for simultaneous movement of multiple unit-sized packages in the modular warehouse have been proposed. While some features are different, the proposed methods consist of three stages. The first stage is the route planning used to perform the movement of each package from the starting location to the destination location. The second stage is the tagging process based on packet priorities, used to prevent packet collisions. The final stage is the main control part where the movement of all packages in the warehouse is controlled.

The proposed methods are compared with the methods of Datar [6] and Sittivijan [8]. Datar [6] chose the next movement area in the route planning section only by looking at the distance to the target and the priorities of the packages. Sittivijan [8] has proposed an A* based heuristic for the movement of packages. The difference between the method proposed in this study and the method proposed by Sittivijan [8] is that the heuristic route planning is carried out at the beginning and subsequent route planning is not carried out as long as the packages do not leave their the planned routes. However, in the proposed method, the route planning process is applied again for the active packages in each environment change. Furthermore, in Sittivijan [8], a release is applied to the object subjected to the tagging process in both the main control and the tagging process. In the proposed method, this operation was removed from the main control section and applied only in the tagging process to ensure achieving high-quality results. Also, an improvement has been made to the conditions of high priority packages to reach their destination at close to their *LB* value calculations during the route planning phase, which provides improved results. In future studies, solution approaches using other heuristics will be developed for warehouse systems with different dimensions.

Acknowledgments

References

[1]   Mason-Jones, R., & Towill, D.R. (1999). Total cycle time compression and the agile supply chain. *International Journal of Production Economics*, 62(1-2), 61-73.

[2]   Çancı, M., & Erdal, M. (2009). *Lojistik Yönetimi: Freight Forwarder El Kitabı*. 3. Baskı, Uluslararası Taşımacılık ve Lojistik Hizmet Üretenleri Derneği, İstanbul.

[3]   Sahin, Y., & Eroğlu, A. (2015). Hierarchical Solution of Order Picking and Capacitated Vehicle Routing Problems. *Suleyman Demirel University Journal of Engineering Sciences and Design*, 3(1), 15-28.

[4]   Xiang, L., Kay, M.G., & Telford, J. (2007). *Public Logistics Network Protocol Design and Implementation* [online]. North Carolina, North Carolina State University, Available from: https://people.engr.ncsu.edu/kay/pln/IERC07.pdf [Accessed 15 August 2018]

[5]   Kay, M. G. (2004). Protocol Design for a Public Logistic Network [online]. North Carolina, North Carolina State University, Available from: https://people.engr.ncsu.edu/kay/pln/IMHRC04.pdf [Accessed 16 August 2018]

[6]   Datar, M. (2011). *Priority-based Control Algorithm for Movement of Packages in a Public Distribution Center*. PhD Thesis, North Carolina State University.

[7]   Kay, M.G. (2013). *Home Delivery Logistics Networks using Driverless Delivery Vehicles [online]*. North Carolina, North Carolina State University, Available from: https://people.engr.ncsu.edu/kay/hdln/HDLNuDDV.pdf [Accessed 15 August 2018]

[8]   Sittivijan, P. (2015). *Modular Warehouse Control: Simultaneous Rectilinear Movement of Multiple Objects within Limited Free Space Environment*. PhD Thesis, North Carolina State University.

[9]   Bauer, B. (1994). *The Manhattan Pair Distance Heuristic for the 15-Puzzle [online]*. Paderborn, Universitat-GH Paderborn. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.7&rep=rep1&type=pdf [Accessed 25 July 2018]

[10]   Spitznagel, E.L. (1967). A new look at the fifteen puzzle. *Mathematics Magazine*, 40(4), 171-174.

[11]   Reinefeld, A. (1993). Complete Solution of the Eight-Puzzle and the Benefit of Node Ordering in IDA*. *Thirteenth International Joint Conference on Artificial Intelligence,* pp 248-253.

[12]   Gue, K.R., & Kim, B.S. (2007). Puzzle-based storage systems. *Naval Research Logistics*, 54(5), 556-567.

[13]   Flake, G.W., & Baum, E.B. (2002). Rush Hour is PSPACE-complete, or "Why you should generously tip parking lot attendants. *Theoretical Computer Science*, 270(1-2), 895-911.

[14]   Hearn, R.A., & Demaine, E.D. (2005). PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2), 72-96.

[15]   Hauptman, A., Elyasaf, A., Sipper, M., & Karmon, A. (2009). GP-Rush: using genetic programming to evolve solvers for the Rush Hour puzzle. *11th Annual Conference on Genetic and Evolutionary Computation*, pp 955-962.

[16]   Sharma, R., & Aloimonos, Y. (1992). Coordinated motion planning: the warehouseman's problem with constraints on free space. *IEEE Transactions on systems, man, and cybernetics*, 22(1), 130-141.

[17]   Hopcroft, J.E., Schwartz, J.T., & Sharir, M. (1984). On the Complexity of Motion Planning for Multiple Independent Objects; PSPACE-Hardness of the Warehouseman's Problem. *The International Journal of Robotics Research*, 3(4), 76-88.

[18]   Yeung, D.Y., & Bekey, G. (1987). A decentralized approach to the motion planning problem for multiple mobile robots. *IEEE International Conference on Robotics and Automation*, 4, 1779-1784.

[19]   Sanchez, G., Latombe, & Jean-Claude (2002). Using a PRM planner to compare centralized and decoupled planning for multi-robot systems. *IEEE International Conference on Robotics and Automation*, 2, 2112-2119.

[20]   Sharma, R., & Aloimonos, Y., (1992). Coordinated motion planning: the warehouseman's problem with constraints on free space. *IEEE Transactions on systems, man, and cybernetics*, 22(1), 130-141.

[21]   Sarrafzadeh, M., & Maddila, S.R. (1995). Discrete warehouse problem. *Theoretical Computer Science*, 140(2), 231-247.

[22]   LaValle, S.M., & Hutchinson, S.A. (1998). Optimal motion planning for multiple robots having independent goals. *IEEE Transactions on Robotics and Automation*, 14(6), 912-925.

[23]   Azarm, K., & Schmidt, G. (1997). Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation. *IEEE International Conference on Robotics and Automation*, 4, 3526-3533.

[24]   Švestka, P., & Overmars, M.H. (1998). Coordinated path planning for multiple robots. *Robotics and Autonomous Systems*, 23(3), 125-152.

[25]   Leroy, S., Laumond, J.P., & Siméon, T. (1999). Multiple path coordination for mobile robots: A geometric algorithm. *16th International Joint Conference on Artificial Intelligence*, pp 1118-1123.

[26]   Guo, Y., & Parker, L.E. (2002). A distributed and optimal motion planning approach for multiple

mobile robots. *IEEE International Conference on Robotics and Automation*, 3, 2612-2619.

[27] Yamashita, A., Arai, T., Ota, J., & Asama, H. (2003). Motion planning of multiple mobile robots for cooperative manipulation and transportation. *IEEE Transactions on Robotics and Automation*, 19(2), 223-237.

[28] Liu, S. Mao, L., & Yu, J., (2006). Path planning based on ant colony algorithm and distributed local navigation for multi-robot systems. *IEEE International Conference on Mechatronics and Automation*, pp 1733-1738.

[29] Koç, Ç., Erbaş, M., & Ozceylan, E. (2018). A rich vehicle routing problem arising in the replenishment of automated teller machines. *An International Journal of Optimization and Control: Theories & Applications (IJOCTA)*, 8(2), 276-287.

[30] Uddin, M.F., & Kazushi, S.A.N.O., (2011). Coordination and Optimization: The integrated supply chain analysis with non-linear price-sensitive demand. *An International Journal of Optimization and Control: Theories & Applications (IJOCTA)*, 2(1), 83-94.

Elif G. Dayıoğlu *received her B.S. and M.S. degree from the Computer Engineering Department of Pamukkale University in 2014 and 2017 respectively. She is currently pursuing a PhD degree in Computer Engineering at Çukurova University. She has been working as a research assistant at the Computer Technology and Infirmation Systems Department of Mersin University since 2014.*

http://orcid.org/0000-0002-4716-952X

Kenan Karagül *studied industrial engineering for his Bachelor degree and business administration for M.S. and PhD degrees. His field of study includes operations research, logistics, vehicle routing problems, metaheuristics, and quantitative models. He worked at various firms between 1996 and 2001. He worked at Pamukkale University as an instructor until 2013. He has been working at the same university as an assistant professor since 2013. He was awarded the best Ph.D. thesis on Graduate Tourism Students Congress in Kuşadası (2014).*

http://orcid.org/0000-0001-5397-4464

Yusuf Şahin *received the M.S. degree in industrial engineering from Pamukkale University in 2009 and a PhD degree in business administration from Suleyman Demirel University in 2014. He has been an assistant professor of business administration at Burdur Mehmet Akif Ersoy University since 2014. His field of study includes operations research, logistics, warehouse management, vehicle routing, meta-heuristics, and quantitative models.*

http://orcid.org/0000-0002-3862-6485

Michael G. Kay *has been a professor of Industrial Engineering at North Carolina State University since 1992. He is Interim Director of the Operations Research Graduate Program and is Associate Director of Graduate Programs in the ISE Department. He is the current President of the College-Industry Council on Material Handling Education.*

http://orcid.org/0000-0002-1359-8270