INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2021, VOL. 67, NO. 4, PP. 559-564
Manuscript received August 26, 2021; revised November, 2021 DOI: 10.24425/ijet.2021.137846

Optimization of Autonomous Agent Routes in Logistics Warehouse

Tomasz Markowski and Piotr Bilski

***Abstract*—The paper introduces the distributed framework for determining the shortest path of robots in the logistic applications, i.e. the warehouse with a swarm of robots cooperating in the Real-Time mode. The proposed solution uses the optimization routine to avoid the downtime and collisions between robots. The presented approach uses the reference model based on Dijkstra, Floyd-Warshall and Bellman-Ford algorithms, which search the path in the weighted undirected graph. Their application in the onboard robot's computer requires the analysis of the time efficiency. Results of comparative simulations for the implemented algorithms are presented. For their evaluation the data sets reflecting actual processes were used. Outcomes of experiments have shown that the tested algorithms are applicable for the logistic purposes, however their ability to operate in the Real-Time requires the detailed analysis.**

***Keywords*—shortest path problem, hive of robots, logistics, microcontrollers**

I. INTRODUCTION

THE market of logistic services, including warehouse applications, has been developing dynamically in recent years. The amount of warehouse space is increasing, so the developers' offers also changes during the competition on the market. Customers, including logistics operators, require modern, flexible storage space, allowing for easy reorganization and facilitating its adaptation to market needs. This trend fits into the usage of automatic and semi-automatic solutions, including autonomous robots and machines. Their purpose is, for instance, to collect clients' orders by picking the selected goods from the particular locations in the warehouse (such as in the Amazon company). Optimization of human and machine tasks is one way to achieve higher resource efficiency and has been developed for years. Currently equipping logistics facilities with advanced transport and completing systems became a new standard. Therefore implementation of automated algorithms for managing warehouses and commodities is important [1] [2] .

Algorithms for the route optimization are known and well documented in the literature (the traveling salesman problem, shortest path algorithms) [5],[6],[7]. However, replacing human workers by their autonomous counterparts, requires transforming the problem of the single route optimization into the task of managing multiple robots in the warehouse space. Optimizing their paths, considering the simultaneous cooperation in the on-line mode should also accept constantly arriving new tasks that must be assigned to the specific agents.

Effective operations' management for up to a dozen of people working in a specific space is to be replaced by the model allowing for the effective work management by the hive of robots. The new goal of optimization is not only minimizing traveled routes in the warehouse, but also rearranging them constantly to avoid collisions, which would cause the unnecessary delays related with stopping the robots and deciding, which one should pass the crossing first.

The following paper introduces the reference model for simultaneous optimization of many robots' paths cooperating in the warehouse. Various types of warehouses were considered, differing in shapes and sizes. The selected group of algorithms (including Dijkstra, Floyd-Warshall and Bellman-Ford) was used to calculate the paths. Accuracy of used methods and their time efficiency were evaluated.

The content of the paper is as follows. In Section II the model of the warehouse is described. Section III introduces the selected path optimization algorithms. In Section IV the implementation details are presented. Section V contains results of simulations. Finally, Section VI covers conclusions and future prospects.

II. MODELING OF STORAGE SPACE

Development of the model for optimization of autonomous robot routes requires computer-based mapping of the warehouse space and tasks to be performed. Literature on the subject reveals the possibility of presenting warehouse space based on a graph and the corresponding neighborhood matrix [8] [9] [10]. Solutions used so far consider only a simple model, in which a single object operates in the warehouse at the specific time instant. Optimization of its route consists in arranging the sequence of discrete steps (taking a single time slice) to minimize the path's length and suppress the time of the completing the assignment. These two aims are not the same and the optimization algorithm should consider both (in the multi-criteria optimization process), or only one of them, depending on the particular needs. The warehouse space mapping considers not only the traveling time, but also the operation duration (by defining an additional, two-dimensional cost matrix). The reference algorithm for the method in this paper is presented on Fig. 1

Its current application is limited to calculations in a single centralized IT system, but the inclusion of routes for many robots requires introducing the parallelism and mutual exchange

Piotr Bilski is with Warsaw University of Technology, Poland (e-mail: pbilski@ire.pw.edu.pl).

Tomasz Markowski is with Lukasiewicz – Institute of Logistics and Warehousing, Poland (e-mail: tomasz.markowski@ ilim.poznan.pl).

of information about the planned route between agents. The model constructed this way will be used to create the distributed reference structure, which exploits the route optimization executed locally in each robot's onboard computer independently. Results of simulations for such a reference model and its distributed extension, by including multiple robots operating in parallel in the warehouse space (using the same, shared list of tasks to perform) will be evaluated in the future.

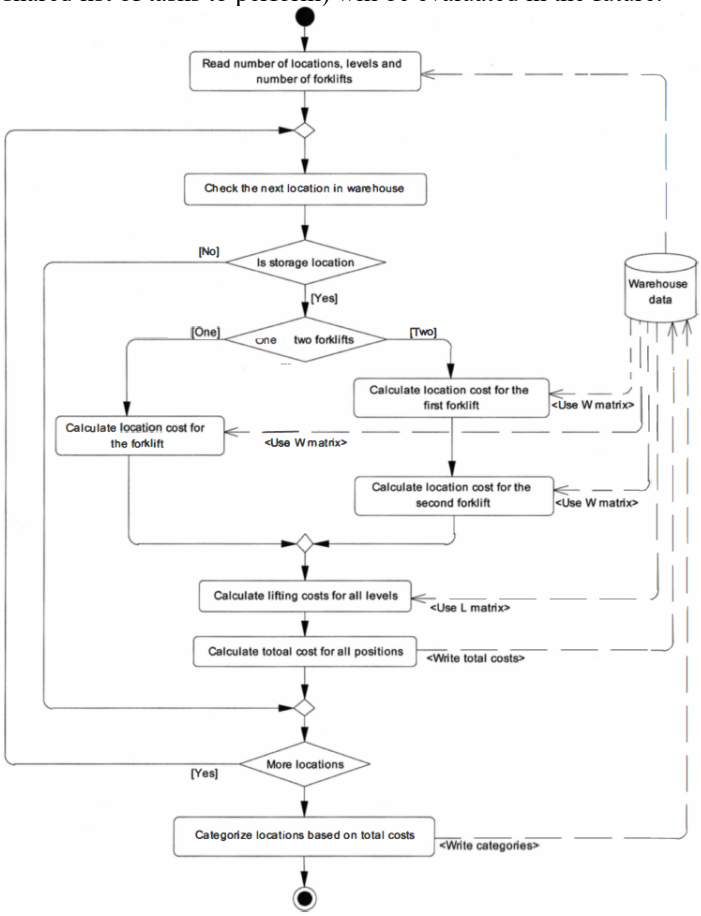

Fig. 1. Single robot path optimization algorithm [8]

III. ALGORITHMS OF ROUTE OPTIMIZATION

The proposed algorithm is the modification of the basic version presented in Fig. 1, which focuses on the standalone agent. The novel approach assumes coexistence of many autonomous robots working concurrently, as presented in Fig. 2. The main difference between the standalone and concurrent version is the need to consider intersections of routes, as well as delays and conflicts resulting from them. The usage of the centralized IT system determining paths for all robots allows for elimination or minimization of possible collisions between routes. In this case, the central computer controls all paths and may counteract accordingly.

Modification of routes requires recalculating all paths, detecting and resolving conflicts. One of the method to suppress the time for calculations is marking nodes as occupied in a specific time instant if the robot located there is pausing (not moving to avoid conflicts). In such a case calculations for this robot may be omitted as long as it stands still.

The proposed distributed algorithm consists of simultaneous calculations by autonomous robots locally in their onboard computers. Knowledge about a mutual influence of other robots working in the same area is limited to a specific number of iterations (referred to as prediction steps) following the currently performed one. The number of prediction steps should be determined experimentally, as each warehouse requires individual algorithm configuration. The proposed algorithm consists of calculating in each iteration the shortest route (step A), considering the location of adjacent robots (step B). Knowledge about their positions is valid for the determining prediction steps. Each robot sends to its counterparts in the vicinity a certain number of planned actions during the incoming iterations, allowing for considering them in their route planning process and, if possible, avoiding collisions (step C). Information about prediction steps fills the neighborhood matrix based on planned fragments of the robots' path. If the specific robot is unable to find a route not interfering with any other robots, it will pause for one iteration (step D).

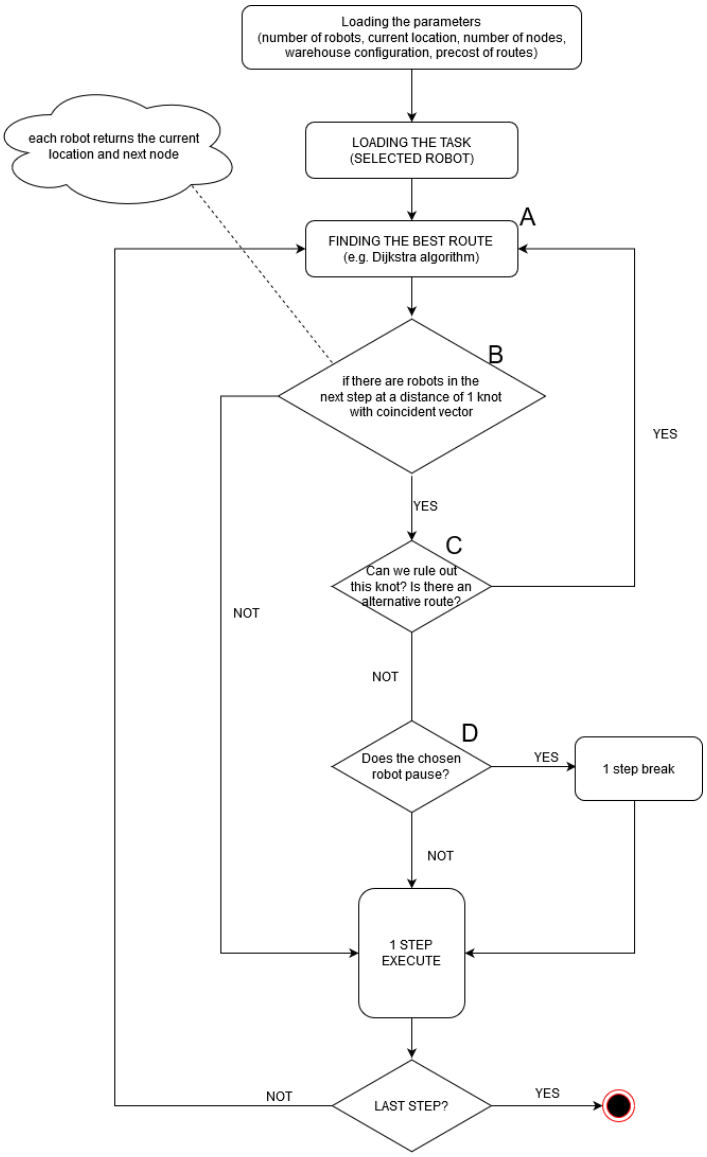

Fig. 2. Proposed multi-agent route optimization algorithm

IV. IMPLEMENTATION

The original single route optimization algorithm was implemented in Python 3 programming language to conduct comparative tests in terms of performance, computational

complexity and efficiency of operation. Python is currently popular because of the simple constructs and abundance of usable libraries. Its disadvantage is relatively low speed (because of the high level of operation on the application stack), making it difficult to use in small computing systems, such as the warehouse robot onboard computer. However, possible embedded implementations exist, such as Raspberry Pi or micropython architectures, implementing the code interpreter in the hardware [3] [4].

To calculate the shortest path between the selected nodes in the graph, Dijkstra, Bellman-Ford and Floyd-Warshall algorithms can be used. They are classical route and flow calculation algorithms applied in weighted graphs with nonnegative weight values. The important problem is their implementation in the specific type of graph, representing the warehouse. [12]

The warehouse model was implemented as a list of vertices' pairs with their transition weights (costs of travelling through them), for instance: [(#v1, #v5, 5), (#v7, #v12, 4), (#v11, #v24, 3)]. The vertices of the graph represent the intersections of the warehouse. It is not necessary to represent each storage location in the model, as the construction of the warehouse makes it impossible to reach the storage location in any other way than through the nearest intersection. On the other hand, replacing storage locations with intersections allows for a significant simplification of the model, which does not affect the correct operation of the algorithm. The edges of the graph exist only for vertices, between which the direct travel is possible (for some adjacent vertices, due to the physical design of the warehouse, the passage is not possible). Duration of the actual operation, understood as the time of completing single task in a given location (e.g. picking up or putting the goods on the shelf) was not taken into account here (though it is not zero).

A. *Types of warehouses*

The input data sets were prepared in a way to reproduce the types of warehouses and storage strategies occurring in the real-world logistics. There are many strategies and methods of placing goods in a warehouse. Their spatial characteristics result primarily from the rotation of goods, sales forecasts or assumed inventory security. Also, indicators defined for a particular enterprise may be relevant (average warehouse turnover, capacity utilization, correctness of warehouse inventory, or technological assessment), as well as share of storage costs, the number of operations per employee and the number of employees. Reproducing all these details in a model is difficult. The easier approach is to create a task list reflecting such details (by choosing task set and order to reproduce selected strategy). In the presented study, two cases were considered. The first one when the goods most often found in particular operations are stored close to each other. The second case assumes their storage location is selected completely at random.

The structure of the graph is the two-dimensional lattice with edges connecting neighboring vertices only vertically and horizontally (so-called grid graph). The rectangular shape limits the number of vertices $|V|$ to $m \cdot n$ (where m and n are numbers of the vertices in both dimensions) and the number of edges $|E|$ to $2 \cdot m \cdot n - m - n$ [7]. In general, the warehouse may have the composite structure being the combination of multiple rectangles, but this case is not considered here.

Simulations for four different warehouse graphs were prepared. The models (later referred to by the roman numerals) differed in complexity, with $|V| \in \{36, 72, 156, 224\}$.

B. *Input data structure*

The definition of logistic operations was implemented as a list of two values, determining the pickup and delivery location. Both are interpreted as addresses in the warehouse from which the commodity is collected and where it is next delivered to. An example list from a CSV file used to describe the location of goods is presented in Fig. 3

```
#v10;#v26
#v5;#v22
#v3;#v12
#v8;#v35
…
#v6;#x20
```

Fig. 3. List of warehouse operations

An analogous data format was used to import warehouse structures. The adjacency matrix was converted into a CSV file allowing for an easy implementation of different graph structure and import. The structure of the sample graph is presented in Fig. 4, where the numbered vertices (starting with the "#v" symbol) are black dots, while horizontal and vertical edges are indicated by integers. The corresponding csv-like representation of this graph is show in Fig. 5.

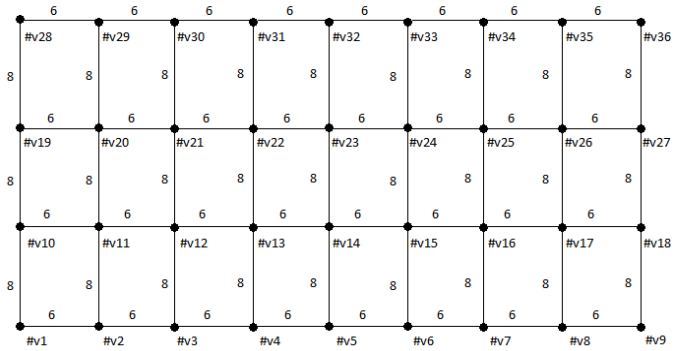

Fig. 4. Example of the considered warehouse graph

#v1	6	#v2	6	#v3	6	#v4	6	#v5	6	#v6	6	#v7	6	#v8	6	#v9
8		8		8		8		8		8		8		8		8
#v10	6	#v11	6	#v12	6	#v13	6	#v14	6	#v15	6	#v16	6	#v17	6	#v18
24		24		24		24		24		24		24		24		24
#v19	6	#v20	6	#v21	6	#v22	6	#v23	6	#v24	6	#v25	6	#v26	6	#v27
8		8		8		8		8		8		8		8		8
#v28	6	#v29	6	#v30	6	#v31	6	#v32	6	#v33	6	#v34	6	#v35	6	#v36

Fig. 5. Warehouse csv-like neighbourhood matrix representation

For the implementation of algorithms some Python build in modules were used: *collection* and *networkx* modules for Dijkstra algorithm, *sys* module for Bellman-Ford and the *math* and *intertools* for the implementation of Floyd-Warshall algorithm.

V. SIMULATIONS

For experiments two reference picking lists have been prepared, consisting of 1000 items, representing warehouse orders. The first list was prepared so that the orders were selected randomly from the entire warehouse area. The second list was created according to the "ABC" principle, in which the warehouse layout, location of goods and thus transport orders are selected

according to the Pareto principle (Fig. 6). This rule indicates three groups of commodities (80%, 15% and 5% of the cumulative value of the examined feature), where the percentage share of assortment items in groups A, B, C in the total number of goods is 20%, 30% and 50% [13] [14] [15].

Both list imported into the simulation were tested on the same warehouse structures: (i) with 36 vertices (4 x 6), (ii) with 72 vertices (8 x 9), (iii) with 156 vertices (12 x 13) and (iv) with 224 vertices (14 x 16). The simulations were carried out on all but also on parts of the list (the first 5, 10, 20, 50, 100, 200, 500 and all 1000 orders respectively).

The aim of simulations was to verify the ability of selected algorithms to perform calculations while the execution time is important due to the large variability of operations performed in the warehouse and recalculating routes in the Real-Time.

All calculations were made using a personal computer with the following configuration: Intel i7 processor (3.2GHz), 8GB RAM and 128SSD hard drive.

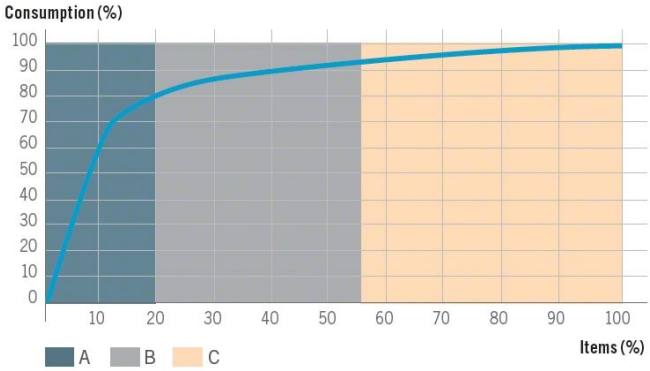

Fig. 6. Graphic representation of a Pareto chart [16]

A. *Simulations for random tasks sequencing*

In the first experiment, the list of random tasks was analyzed. All algorithms were tested on the list and graph A. Results are presented in Fig. 8, from which the computational complexity can be evaluated.

The algorithms' work regimes explain the obtained results - the calculation complexity for the Floyd-Warshall algorithm does not change for increasing numbers of tasks, for the Bellman-Ford algorithm the simulation time increases by approximation with Q^2 (where Q is the tasks quantity), while for the Dijkstra algorithm the increase in the time of determining the shortest routes also by approximation increases exponentially with the number of tasks.

The subsequent experiments were aimed at determining the impact of the warehouse size on time of computations. Simulations were repeated successively for all warehouses (A, B, C, D) and various task sets. Obtained results are presented below. In Fig. 8 the Floyd-Warshall algorithm simulation results are shown. Fig. 9 presents simulation results for Bellman-Ford algorithm. Finally, in Fig. 10 Dijkstra algorithm simulation results are presented.

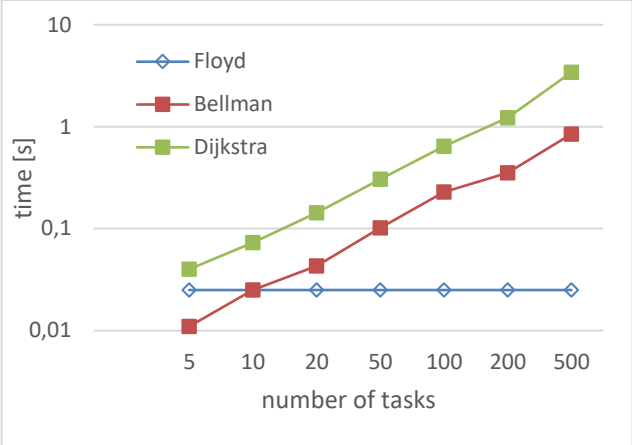

Fig. 7. Algorithms efficiency in the graph (i)

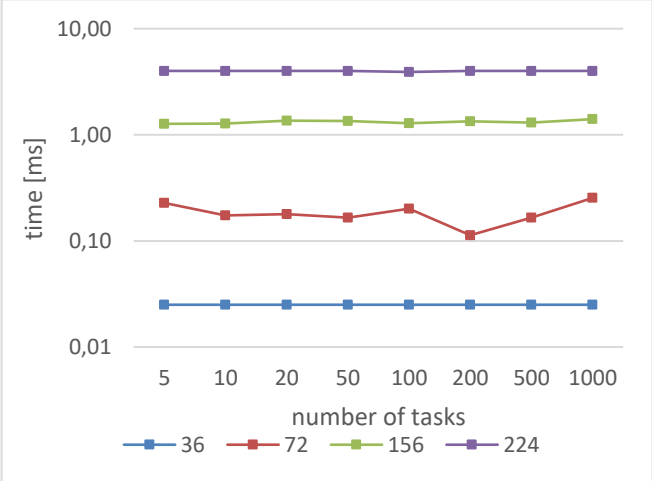

Fig. 8. Floyd-Warshall algorithm performance

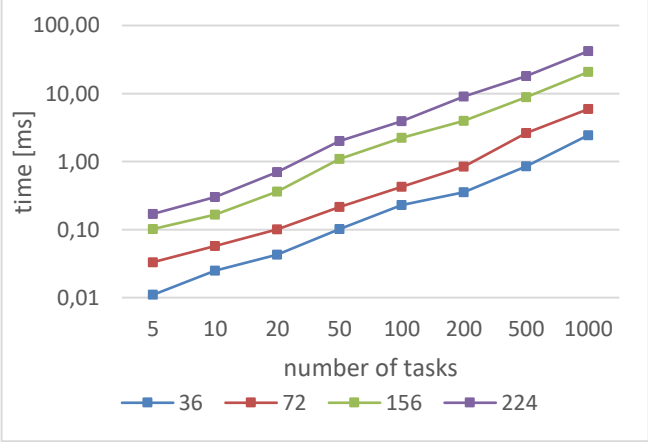

Fig. 9. Bellman-Ford algorithm performance

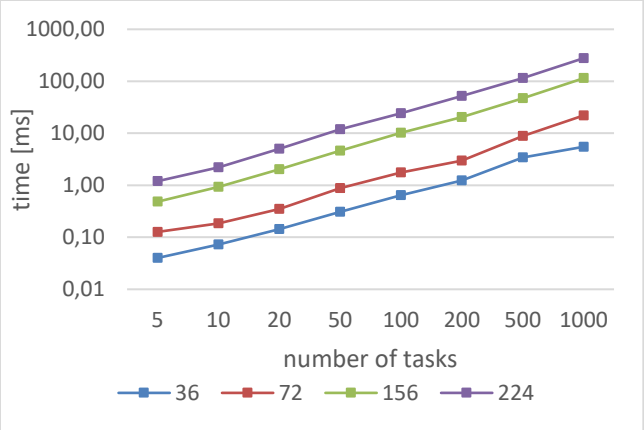

Fig. 10. Dijkstra algorithm performance

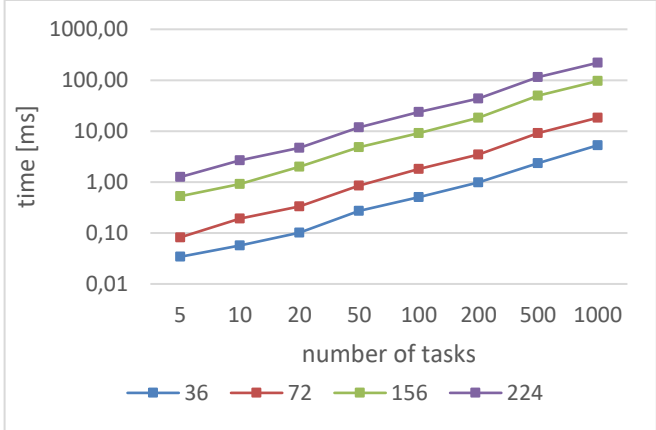

Fig. 13. Dijkstra algorithm performance

B. ABC tasks simulations

In the second part of experiments the list consisting of tasks constructed with the ABC rules was used. In Fig. 11 Floyd-Warshall algorithm simulation results on ABC tasks are presented. In Fig. 12 Bellman-Ford algorithm simulation results on ABC tasks are shown. Fig. 13 contains results for Dijkstra algorithm for ABC tasks. The conducted experiments focused on comparing the time required to determine the optimal path. Under this assumption, the way the goods were arranged (random or ABC-compliant) in the warehouse was irrelevant to the obtained results, since the calculations were not meant to depict the picking time of these goods, so the obtained times of calculation were, as expected, analogous.

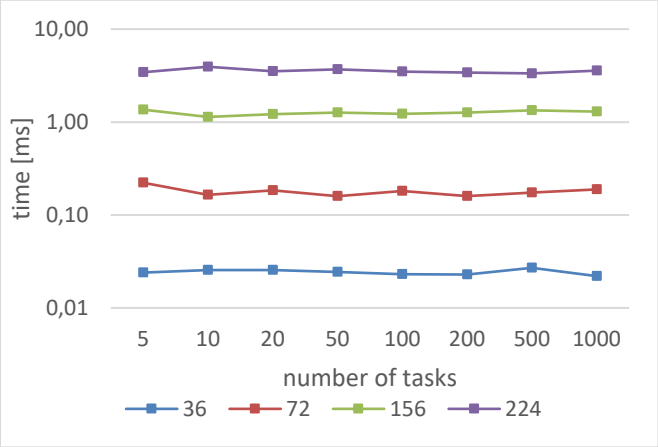

Fig. 11. Fig. 8. Floyd-Warshall algorithm performance

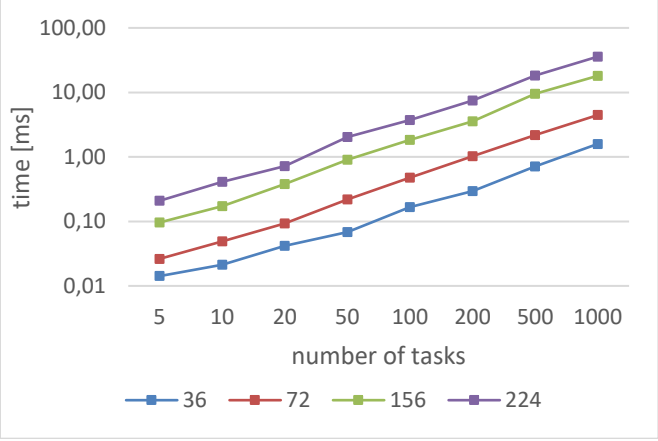

Fig. 12. Bellman-Ford algorithm performance

VI. CONCLUSIONS

Presented simulations included route mapping for a varying number of picking tasks - from 5 to 1000, and three different algorithms for determining the shortest path implemented: the Floyd-Warshall, Bellman-Ford and Dijkstra algorithms. An improved way of data preparation has been developed - in the form of a CSV file, facilitating the introduction of larger warehouse structures, and a greater number of picking orders (pairs of vertices between which a path should be marked). Structures for simulations were warehouses / graphs containing 36 to 224 vertices, and order lists containing 5 to 1000 orders.

The prepared order lists reflected the adopted storage strategies - completely random, without an ordered storage structure, and ordered by ABC strategy. Lists of orders were built based on vertices drawn from areas consistent with the adopted strategy.

The simulation results compare several cases - dependence of obtained results on:

- the size of the warehouse
- the number of orders
- the number of agents / robots
- the adopted picking strategy.

The simulations results indicated that in the case of calculating more paths in one run, the most effective algorithm is the Floyd-Warshall algorithm. However, for the calculation of single track changes, the fastest calculation time was obtained for the Dijkstra algorithm. Therefore, further implementations will be based on a combination of the use of these algorithms and not one selected algorithm.

VII. FUTURE WORK

An ESP32 platform was prepared for the implementation of the abovementioned algorithms (ESP32-wroomer system) – Fig. 14.

Fig. 14. ESP32 Board

The implementation is developed in micropython, and the further development of the algorithm is planned.

Based on the obtained simulation results, the algorithm will be modified according to which the paths in the distributed model (in the agent network) will be calculated in order to better use the capabilities of each algorithm.

REFERENCES

[1] Mobile Robot Platforms, Shuttle Automated Storage and Retrieval Systems, Industrial Robotic Manipulators, and Gantry Robots: Global Market Analysis and Forecasts, Informa PLC, https://www.tractica.com/research/warehousing-and-logistics-robots/

[2] J. Miklinska, "Trends in the logistic market and warehouses for logistics service providers-experiences from Poland," *Economic and Social Development: Book of Proceedings*, 2020, 193-202.

[3] M. Khamphroo, N. Kwankeo, K. Kaemarungsi, K. Fukawa, "MicroPython-based educational mobile robot for computer coding learning," 2017 8th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES), Chonburi, 2017.

[4] K. Dokic, B. Radisic, M. Cobović, "MicroPython or Arduino C for ESP32 - Efficiency for Neural Network Edge Devices," *Springier*, 2020, pp.33-34, https://doi.org/10.1007/978-3-030-43364-2_4.

[5] N. Deo, "Graph theory with applications to engineering and computer science," *Englewood Cliffs, NJ: Prentice-Hall*, 1974.

[6] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *EJOR*, 1992, Vol.59, pp. 231-247.

[7] Lu Feng, "Shortest path algorithm: Taxonomy and Advance in Research", *Acta Geodaetica et Car tographica Sinica*, vol. 30, no. 3, pp. 269-275, 2001.

[8] D. Dobrilovic, V. Jevtic, I. Beker, Z. Stojanov, "Shortest-path based Model for Warehouse Inner Transportation Optimization" in 7th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)

[9] Y. Liu, T. M. Vitolo, "Graph Data Warehouse: Steps to Integrating Graph Databases Into the Traditional Conceptual Structure of a Data Warehouse," *2013 IEEE International Congress on Big Data*, 2013, pp. 433-434, https://doi.org/10.1109/BigData.Congress.2013.72

[10] H.Y. Jang, J.U. Sun, "A Graph Optimization Algorithm for Warehouses with Middle Cross Aisles," *Applied Mechanics and Materials*, 2011, 145. 354-358, https://doi.org/10.4028/www.scientific.net/AMM.145.354.

[11] B.D. Acharya, M.K. Gill, "On the Index of Gracefulness of a Graph and the Gracefulness of Two-Dimensional Square Lattice Graphs, " *Indian J. Math.*, 1981, 23, 81-94.

[12] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, "Introduction to algorithms," MIT Press, 1994.

[13] Warehouse material flows and flow charts, https://www.mecalux.co.uk/warehouse-manual/warehouse-design/warehouse-material-flowchart

[14] A. Niemczyk et al., "Organizacja i monitorowanie procesów magazynowych," Instytut Logistyki i Magazynowania, 2014.

[15] A. Szymonik, D. Chudzik, "Logistyka nowoczesnej gospodarki magazynowej," Difin, 2018.

[16] B. Mbakop A. Kevine, "The Effectiveness of ABC Cross Analysis on Products Allocation in the Warehouse," 2018, January – February, Vol. 5, Issue 1, pp: 11-30.