International Journal of Industrial Engineering and Management (IJIEM), Vol.3 No 1, 2012, pp. 39-45
Available online at http:// www.iim.ftn.uns.ac.rs/ijiem_journal.php
ISSN 2217-2661

**UDK: 005.591.1:621.868.27
004.65**

Research paper

Shortest-path algorithms as a tools for inner transportation optimization

Ivan Beker
University of Novi Sad/Faculty of Technical Sciences, Novi Sad, Serbia
Vesna Jevtić
University of Novi Sad/Technical Faculty “Mihajlo Pupin”, Zrenjanin, Serbia
Dalibor Dobrilović
University of Novi Sad/Technical Faculty “Mihajlo Pupin”, Zrenjanin, Serbia

Received (15 December 2011.); Revised (24 January 2012.); Accepted (08 February 2012.)

Abstract

This paper deals with business processes and productivity improvement in order to reduce costs of any kind, especially by eliminating of the wastes. Therefore, automation of logistic processes is very important, and in this case is the source of reducing one of the biggest wastes of all: inner transportation. There are several models of routes selecting in practice and, the main focus of this paper is to investigate the implementation of one of them - the shortest path algorithms for forklifts routing optimization. By calculating the optimal route for forklifts, transportation routes are shortened and work in the warehouses is reduced. These algorithms can be applied for the other types of vehicles and for the other type of storage facilities as well. Preview of the optimization methods is used for identification of the most suitable method.

Key words: *logistics costs, forklift route, optimization, Dijkstra algorithm*

1. INTRODUCTION

Today's economy demand very serious commitment to eliminating of the waste of every kind. Maybe the first serious work in this field is done by Toyota and their Production System (TPS), also well known under the term "lean production". Toyota identified 7 deadly wastes: overproduction, unnecessary inventory, unnecessary motion, waiting, transportation, over-processing, scrap and rework. Some of these wastes can be easily identified, but some of them can be accepted as necessary. Logistics is the source of such kind of the, probably, the most evil waste of all: inner transportation.
A 1988 study in the United Kingdom revealed that 55% of all operating costs in a typical warehouse can be attributed to order picking [1], and this fact makes inner transportation as a prime candidate for optimization.
If one considers the piece part production in a factory, for instance, then, usually, each department has its own means of transportation. Trucks or forklifts are sent out to get material from other departments and also to bring finished goods to stores. Normally the transportation capacity is empty one way of the trip, resulting in a capacity utilization of around 50%. This is due to the fact, that the one department does not know of the transportation requirements of other departments. If the control of all means of transportation is centralized then two possible scenarios exist: Either the material flows are that steady (in direction and quantity) that fixed routes can be established on which transportation is carried out regularly. Or this is not possible; then the control centre faces a very serious problem: every point in factory can be a well and a sink at the same time. In addition, demands for transportation occur irregularly and with different urgency and the scheduling cannot be done in advance, but has to be executed on-line. Hence many assignment and scheduling problems have to be solved simultaneously and very fast. With modem hardware and software and using a combination of modified traditional Operations Research algorithms and heuristics this task can be solved [2].
Savings that can be made by optimization of the routes can be significant. According to Djukic et al [3], with routing order-pickers efficiently using routing methods it is possible to achieve a reduction between 17 and 34% in travelling distance. The amount of reduction depends on the particular method used. Although many algorithms for an optimal route have been invented long time ago, in practice heuristics are predominantly used to route the forklift trucks. So, in the practice, different models of routes selecting can be found (Figure 1).
The solution of the problem that involves determining of an optimal route from one point to another is needed in many different areas. The model of the problem is cyclic directed graph diagram whose nodes (vertices) represent points (or cities) and the branches (weights),

i.e. links between them, represent the time required to travel from one to another or the cost of that travel. Determining the optimal path, that will provide minimal time for travelling from the point *i*-th to the point *j*-th, can be achieved by several techniques.

Very popular and widely used are shortest path algorithms, especially in transport problems, as well as in computer networks. There are algorithms such as: Dijkstra's, A*, Floyd-Warshall's and Bellman-Ford's. Then, Routing Information Protocol (RIP) is a dynamic routing protocol based on Bellman-Ford distance vector algorithm [4, 5]. Open Shortest Path First (OSPF) and OSI protocol IS-IS, are dynamic routing protocols too, and they are using Dijkstra shortest path algorithm [6, 7, 8, 9].

The main focus of this paper is to investigate the implementation of the shortest path algorithms for forklifts routing optimization, in order to improve work in the warehouses. Furthermore, these algorithms can be applied to the other types of vehicles and for the other type of storage facilities as well. In that way, business processes and productivity can be improved, as well as, costs reduced and automation of logistic processes achieved.

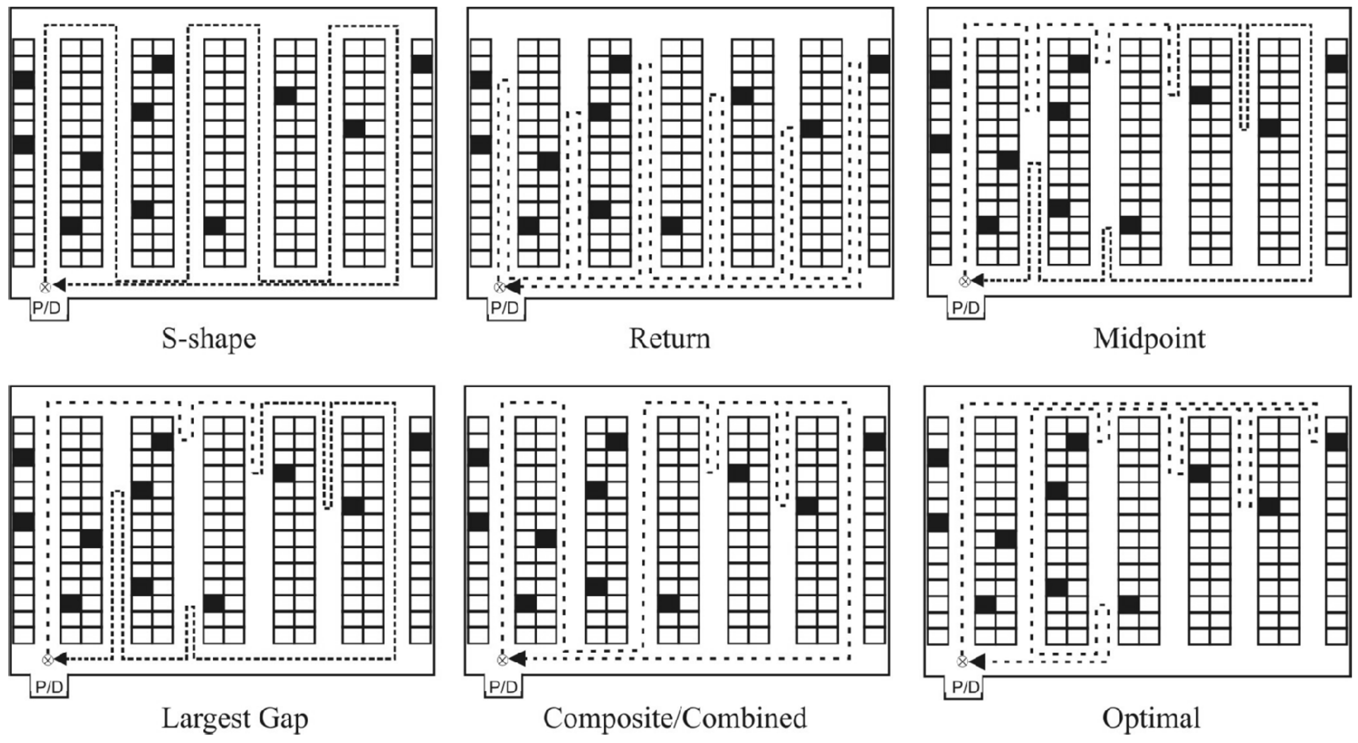


Figure 1. Different models of routes selecting

2. DESCRIPTION OF SHORTEST PATH ALGORITHMS

There are several shortest path algorithms that can be found in the literature [9, 10, 11]. One of them and probably the best known is Dijkstra algorithm (discovered by E.W. Dijkstra [9]). This algorithm solves the single source shortest path problem for a graph with non-negative weights. From a starting vertex, Dijkstra algorithm finds the shortest path to all other vertices in a graph.

The pseudo code of this algorithm is presented in the Figure 2.

In order to increase the speed of Dijkstra's algorithm, A* algorithm is introduced. This algorithm is similar to the previous one and uses heuristic for better time performance. Its heuristic controls choosing the node that is "most likely" to lead to the shortest overall path. A* algorithm is efficient only in case if heuristic is accurate, otherwise it is slower then Dijkstra's algorithm.

In more detail, A* works in iterations. At each iteration, it considers one node of the graph and follows its outgoing connections. The node (called the "current node") is chosen using a selection algorithm similar to Dijkstra's, but with the significant difference of the heuristic. [13]

The pseudo code of A* algorithm is presented in the Figure 3.

More generalization of the problem solving can be achieved by Floyd algorithm. Its improvement in speed is enabled with Floyd-Warshall algorithm. This algorithm finds shortest paths between all pairs of vertices in a directed graph with arbitrary edge weights, although, the negative cycles are forbidden [14, 15].

The pseudo code of Floyd-Warshall algorithm is presented in Figure 4.

Another running time improvement of Dijkstra algorithm is achieved in Bellman-Ford algorithm. This algorithm finds the shortest paths from a source vertex to all the other vertices, but, unlike the previous one, Bellman-Ford algorithm is usually used when there are negative edge weights (negative cycles are not allowed) [15]. The two algorithms differ in the network element on which they iterate. Dijkstra's algorithm iterates on the length of the path (updates it in every iteration) whereas Bellman-Ford's algorithm iterates on the number of edges in the path [13]. The pseudo code of Bellman-Ford algorithm is presented in Figure 5.

```
procedure SPD (G,s,t)
S:=0;
L(s):=0;
for v  V \ {s} do L(v) := ∞;
      while t  S do                {while S < V do}
          begin
          v:= argmin {L(v): v  S};          {S'=V\S}
          S:=  S∪ {v};
          for w  N {v}  ∩ S' do
            if L(w) > L(v) + $l_{vw}$ then
                          begin L(w) := L(v) + $l_{vw}$ ; P(w) := v end
          end;
      P := ⟨t⟩; v := t;
      repeat
            v:= P(v);
            P := v  P;
      until v = s;
      SPD:=P;
end {procedure}
```

Figure 2. Shortest path Dijkstra [12]

```
function A* (start,goal)
closedset:=the empty set; {The set of nodes already evaluated}
openset:= set containing the initial node; {The set of tentative
nodes to be evaluated}
camefrom:=the emtpy map; {The map of navigated nodes}
g_score[start]:= 0 {Cost from start along best known path}
h_score[start]:= heuristic_cost_estimate(start, goal)
f_score[start]:= h_score[start] {Estimated total cost from start to
goal through y}
   while openset <>0 do
        x:= the node in openset having the lowest f_score[] value
      if x = goal
         return reconstruct_path(came_from, came_from[goal])
         remove x from openset
         add x to closedset
      foreach y in neighbor_nodes(x)
         if y in closedset
            continue
         tentative_g_score := g_score[x] + dist_between(x,y)
          if y not in openset
            add y to openset
            tentative_is_better := true
         else if tentative_g_score < g_score[y]
            tentative_is_better := true
         else
            tentative_is_better := false
          if tentative_is_better = true
            came_from[y] := x
            g_score[y] := tentative_g_score
            h_score[y] := heuristic_cost_estimate(y, goal)
            f_score[y] := g_score[y] + h_score[y]
         return failure
        function reconstruct_path(came_from, current_node)
      if came_from[current_node] is set
      p = reconstruct_path(came_from, came_from[current_node])
      return (p + current_node)
   else
      return current_node
end {function}
```

Figure 3. A* algorithm [12]

```
for i := 1 to N
  for j := 1 to N
    if there is an edge from i to j
      dist[0,i,j] := the length of the edge from i to j
   else
   dist[0,i,j] := INFINITY
   end
end
for k := 1 to N
 for i := 1 to N
   for j := 1 to N
   dist[k,i,j] := min{dist[k-1,i,j], dist[k-1,i,k]+dist[k-1,k,j]}
   end
 end
end
```

Figure 4. Floyd-Warshall algortihm

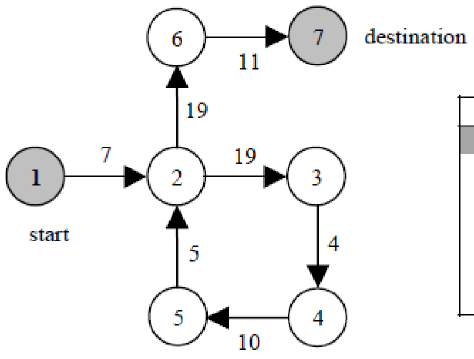
```
procedure BF (int(s))
      int i, j;
      for i := 0 to n – 1
            d[i] := INFINITY;
      d[s] := 0;
      for i := 0 to n – 2
            for j := 0 to e – 1
                  if (d[edges[j].u] + edges[j].w < d[edges[j].v])
                        d[edges[j].v] = d[edges[j].u] + edges[j].w;
                              end {procedure}
```

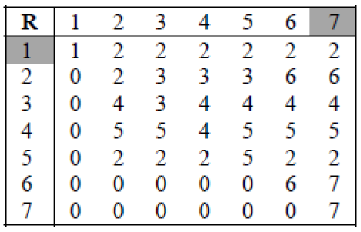
Figure 5. Bellman-Ford algorithm

In correlation with the particular problem's characteristics, one or more of the abovementioned algorithms can be used.

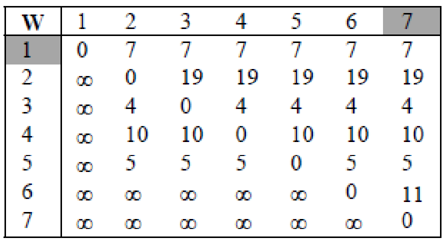
3. EXISTING SOLUTIONS

Above-mentioned algorithms are used for path planning in several researches. For example, Hentschel et al [16] made application for trajectory generation for the autonomous RTS-STILL robotic forklift truck which is able to localize and navigate freely in dynamic environment. They used a graph-based routing algorithm for combination of different routes and finding the shortest path between any two vertices. The graph $G = (V, E)$, is defined by a set of vertices V and a set of edges E whereas the vertices $v_a \in V$ and $v_b \in V$ are connected by the edge $e_{a,b} \in E$. As presented by Floyd (1962) [10], a weight matrix *W* is used for computing the minimum costs among all paths between two vertices. The element $w_{a,b} \in W$ comprises the costs for that particular path. In addition, a route matrix R is introduced for storing the shortest paths in G. Both matrices are initialized by inserting the directed edges $e_{a,b}$ in the route matrix R and the corresponding costs weight $e_{a,b}$ into matrix W (Figure 6.) [16]





R	1	2	3	4	5	6	7
1	1	2	2	2	2	2	2
2	0	2	3	3	3	6	6
3	0	4	3	4	4	4	4
4	0	5	5	4	5	5	5
5	0	2	2	2	5	2	2
6	0	0	0	0	0	6	7
7	0	0	0	0	0	0	7



W	1	2	3	4	5	6	7
1	0	7	7	7	7	7	7
2	∞	0	19	19	19	19	19
3	∞	4	0	4	4	4	4
4	∞	10	10	0	10	10	10
5	∞	5	5	5	0	5	5
6	∞	∞	∞	∞	∞	0	11
7	∞	∞	∞	∞	∞	∞	0

Figure 6. R and W matrices for the example network on the left [16]

Furthermore, Vivaldini et al [17] presented routing system applied on intelligent warehouses, which is able to solve traffic jams and collisions, generate conflict-free and optimized paths before sending the final paths to the robotic forklifts. Among others, their algorithm is based on Dijkstra's shortest path algorithm.

They also proposed a software architecture that considers the local and global tasks of the robotic forklifts (e.g.: local navigation, obstacle avoidance, and auto-localization), that is presented on Figure 7 [17].

Also, Vivaldini et al used the warehouse model that is composed of a fleet of six forklifts that move in a bi-directional circuit composed by 360 nodes, interconnected by 652 arcs as shown in Figure 8. There are some stations: 6 Depot stations for forklift robots, 4 production stations, 11 Shelves with various stations, and 6 Charging Platform stations. They considered that the Unit Control Central (UCC) calculates the routes using the routing algorithm and sends to the forklift robots [18].

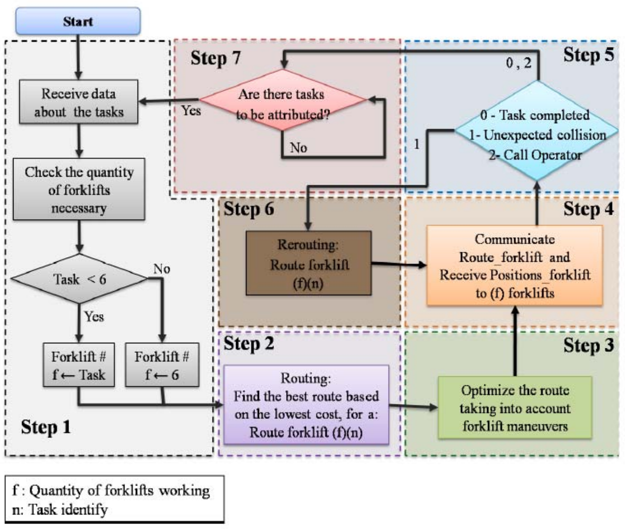

Start
Receive data about the tasks
Check the quantity of forklifts necessary
Task < 6
No
Yes
Forklift # f ← Task
Forklift # f ← 6
Step 1
Step 7
Are there tasks to be attributed?
Yes
No
Step 5
0 , 2
1
0 - Task completed
1- Unexpected collision
2- Call Operator
Step 6
Rerouting: Route forklift (f)(n)
Step 4
Communicate Route_forklift and Receive Positions_forklift to (f) forklifts
Step 2
Routing: Find the best route based on the lowest cost, for a: Route forklift (f)(n)
Step 3
Optimize the route taking into account forklift maneuvers
f : Quantity of forklifts working
n: Task identify

Figure 7. Proposed software architecture [17]

They used Dijkstra's shortest path to calculate the routes of the forklifts, considering the cost.

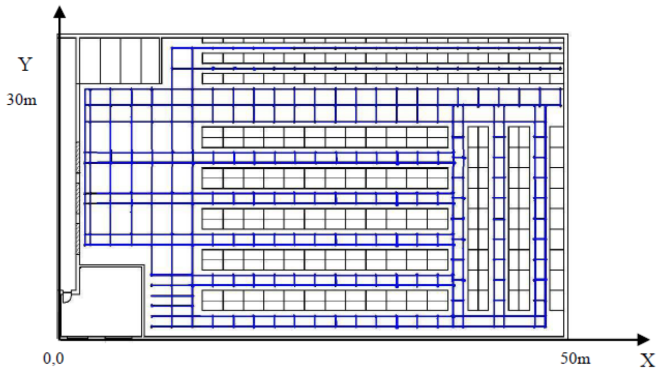


Figure 8. The final graph of the mapping model [18]

IV. PROPOSED MODEL

This paper presents the model for optimization of item location in the warehouse. It is assumed that in certain moments some locations in the warehouse are empty. In order to optimize location of the new items, Dijkstra algorithm calculates path to the most appropriate one. Also, it is assumed that some items have higher priority, i.e. they are used more frequently. For that reason they have to be placed in the most accessible place (the most appropriate one according to item priority level). Example warehouse from the Figure 8 is used in order to experiment with Dijsktra algorithm implementation and the result is given in the Figure 9.

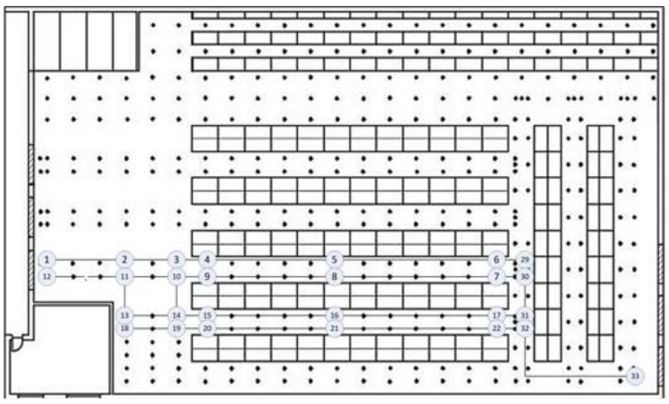


Figure 9. Example warehouse with graph diagram for particular locations

This warehouse has two idle forklift locations (nodes 1 and 33), the first one for items loading and the last one for items discharge. Therefore, item location is calculated after route optimization from the both locations, because every part has both paths to pass, first on enter to warehouse and second when leaving the warehouse. The corresponding cyclic graph diagram with path costs is presented in the Figure 10.

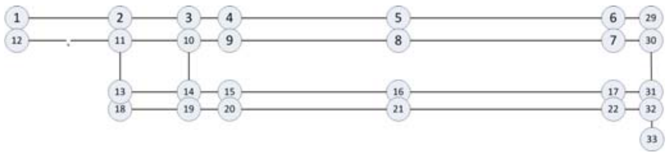


Figure 10. The corresponding cyclic graph diagram with path costs.

In order to extend model [19] usage on warehouses with three-dimensional (3D) storage locations L matrix is introduced. Matrix L(i, j), (i=1,m, j=1,n) represents item position in 3D warehouse, where are: m – the number of the nodes in the graph (locations), and n – the number of the levels (shelves) at the i-th location (node). Extension of the previous warehouse model (Figure 10) with its 3D part is presented in the figure 11.

The path cost corresponds to the time needed for forklift to pass between two neighbouring locations (nodes).

In order to present the graph, and to prepare its data to acceptable input format for the calculation, W matrix (Table I) is introduced. Elements of the W matrix present the path cost between the nodes. In the same time they present connectivity between the nodes. The non-zero values mean that connectivity exists.

Table I Path weight and connectivity matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	∞	5	∞	∞	∞	∞	∞	∞	∞	∞	∞	1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	5	∞	4	∞	∞	∞	∞	∞	∞	∞	1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
3	∞	4	∞	2	∞	∞	∞	∞	∞	1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
4	∞	∞	2	∞	6	∞	∞	∞	1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	6	∞	6	∞	1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	6	∞	6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	2	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	6	∞	6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	2	∞	∞	∞
8	∞	∞	∞	∞	1	∞	6	∞	6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	1	∞	∞	∞	6	∞	2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	1	∞	∞	∞	∞	∞	2	∞	4	∞	∞	3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
11	∞	1	∞	∞	∞	∞	∞	∞	∞	4	∞	5	3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
12	1	∞	∞	∞	∞	∞	∞	∞	∞	∞	5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
13	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	3	∞	∞	4	∞	∞	∞	1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
14	∞	∞	∞	∞	∞	∞	∞	∞	∞	3	∞	∞	4	∞	2	∞	∞	∞	1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
15	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	2	∞	6	∞	∞	∞	1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
16	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	6	∞	6	∞	∞	∞	1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
17	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	6	∞	∞	∞	∞	∞	1	∞	∞	∞	∞	∞	∞	∞	∞	2	∞	∞
18	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	1	∞	∞	∞	∞	∞	4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
19	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	1	∞	∞	∞	4	∞	2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
20	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	1	∞	∞	∞	2	∞	6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
21	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	1	∞	∞	∞	6	∞	6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
22	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	1	∞	∞	∞	6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
23	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	2	∞
24	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
25	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
26	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
27	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
28	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
29	∞	∞	∞	∞	∞	2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	1	∞	∞	∞
30	∞	∞	∞	∞	∞	∞	2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	1	∞	3	∞	∞
31	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	3	∞	1	∞
32	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	2	∞	∞	∞	∞	∞	∞	∞	∞	1	∞	9
33	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	9	∞

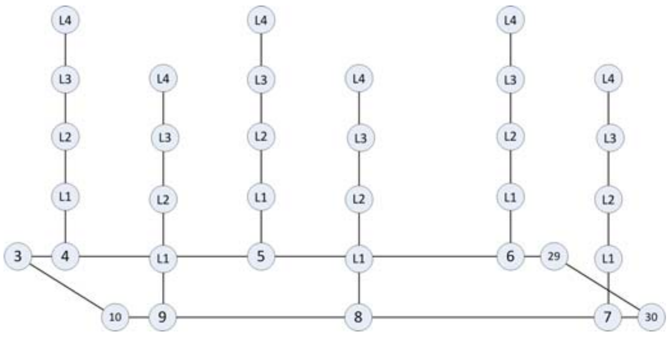


Figure 11. 3D part of the previous warehouse model

This example describes part of the L matrix given in the Table II.

In this example, nodes (locations in the warehouse) number 3 and number 10 are not located in front of the storage and therefore they have zero values in corresponding column. Locations 4, 5 and 6 are in front of three-level shelve, locations 7, 8 or 9 are in front of the four-level shelve. Assigned value to the path cost for item lifting on each level is 1.

Table II Part of L matrix for the 3D warehouse model

i \ j	3	4	5	6	7	8	9	10
1	0	1	1	1	1	1	1	0
2	0	1	1	1	1	1	1	0
3	0	1	1	1	1	1	1	0
4	0	0	0	0	1	1	1	0

The pseudo code for the main procedure of application for item location is given in the Figure 12. This main procedure calls SPD procure (Figure 2.) to calculate path cost and the shortest path for every given empty location starting from locations (nodes) 1 and 33.

```
procedure main
Initialize locations; {add empty locations in the vector}
Initialize levels; {add empty levels at locations in the vector}
n:=Dim {locations};
m:=Dim {levels};
for i = 1 to n
      begin
         SDP(W,1,i);
         SDP(W,33,i);
         print i; {print node i};
         print Lo; {print the output path cost for node i};
         print Li; {print the input path cost for node i};

        hight=0; { calculate and add level to the cost}
        j=levels(i);
            for m=1 to j
            hight=hight+L(m,j);
            end
      loutputcost=outputcost+hight;
      linputcost=inputcost+hight;
      totalcost= loutputcost+ linputcost;

      print loutputcost; { output cost to node i from 1 with level j};
      print linputcost; {intput cost to node i from 33 with level j};
      print totalcost; {total cost for location i };

     print Po; {print the output path to node i from node 1};
     print Pi; {print the input path to node i from node 1};

    end
end
end
```

Figure 12. Pseudo code for Item location optimization

The following input data are used for modified Dijkstra algorithm presentation:

```
location = [7 9 17 22];
level=[1 3 4 2];
```

Those data represent empty location in the warehouse, i.e. location (node) 7 at level 1, location (node) 9 at level 3, etc. are empty. Dijkstra algorithm calculates both paths, the path between node 1 (output-cost) or node 33 (input-cost) and chosen empty locations or locations to be emptied. Total cost includes both path length and the time needed for item lifting. For example, node 7 has output-cost of 24 and input-cost of 15, as well as lifting output-cost of 25 and lifting input-cost of 16. In that case total path cost for node 7 is 41 (Figure 13). Because of the lowest path cost this node is reserved for the items with the highest priority, and that would be item with highest manipulating frequency.

6. CONCLUSION AND FURTHER WORK

Today's economy recognized inner transportation as the source for one of the biggest wastes of all. The fact that 55% of all operating costs in a typical warehouse come from order picking, makes inner transportation as a prime candidate for optimization. There are several models of routes selecting in practice and, the main focus of this paper was to investigate the implementation of the shortest path algorithms for forklifts routing optimization.

Short description and presentation of: Dijkstra, A*, Floyd-Warshall and Bellman-Ford algorithm is given in this paper, as well as a preview of their usage for path planning. First, research results of Hentschel et al was application for trajectory generation for the autonomous forklift truck in dynamic environment. Second, Vivaldini et al presented routing system applied on intelligent warehouses, and they also proposed a software architecture that considers the local and global tasks of the robotic forklifts.

```
node: 7
output-cost: 24
input-cost: 15
lifting output-cost: 25
lifting input-cost: 16

total cost: 41

output path = 1   12   11   10    9    8    7
input path = 33    32    31    30    7

-----

node: 9
output-cost: 12
input-cost: 27
lifting output-cost: 15
lifting input-cost: 30

total cost: 45

output path = 1   12   11   10    9
input path = 33   32   31   30    7    8    9

-----

node: 17
output-cost: 27
input-cost: 12
lifting output-cost: 31
lifting input-cost: 16

total cost: 47

output path = 1 12 11 13 14 15 16 17
input path = 33    32    31    17

-----

node: 22
output-cost: 28
input-cost: 11
lifting output-cost: 30
lifting input-cost: 13

total cost: 43

output path = 1 12 11 13 18 19 20 21 22
input path = 33   32   22
```

Figure 13. Dijkstra path cost calculation

The optimization that is reconsidered in this paper referred to the new items location in the warehouse, according to their priority level. Example warehouse is used in order to experiment with Dijkstra algorithm implementation on the proposed model. The model covers three-dimensional storage and calculated path includes the path between idle nodes and empty location and the time needed for item lifting. Experiment showed the cost path for chosen locations in the warehouse, as well as the nodes on the shortest path. According to the results the location with the minimum path cost can be selected for the item with the highest priority level at the moment.

The model based on one of the shortest path algorithms, such as the one presented in this paper, can be applied not only for forklifts routing optimization, but to the other types of vehicles and for the other type of storage facilities as well. In that way, business processes and productivity can be improved, costs reduced and automation of logistic processes achieved.

ACKNOWLEDGMENT

This research is financial supported by Ministry of Science and Technologic Development, Republic of Serbia; under the project number TR32044 “The development of software tools for business process analysis and improvement”, 2011-2014.

7. REFERENCES

[1] U.S.S. Dharmapriya, A.K.Kulatunga, New Strategy for Warehouse Optimization – Lean warehousing, Proceedings of the 2011 International Conference on Industrial Engineering and Operations Management Kuala Lumpur, Malaysia, January 22 – 24, 2011

[2] H.J. Zimmermann, Applications of intelligent systems in Transportation logistics, INTELLIGENT DECISION MAKING SYSTEMS, Proceedings of the 4th International ISKE Conference on Intelligent Systems and Knowledge Engineering, Hasselt, Belgium, 27 - 28 November 2009

[3] Goran Đukić, Vedran Česnik and Tihomir Opetuk, Order-picking Methods and Technologies for Greener Warehousing, Strojarstvo 52, pp 23-31, 2010

[4] G. Malkin, RFC 2453, RIP version 2, The Internet Society, November 1998.

[5] Richard Bellman: On a Routing Problem, in Quarterly of Applied Mathematics, 16(1), pp. 87–90, 1958.

[6] J. Moy, RFC 2328, OSPF version 2, The Internet Society, April 1998.

[7] R. Coltun, D. Ferguson, J. Moy, A. Lindem, RFC 5340, The Internet Society, July 2008.

[8] D. Oran, RFC 1142, The Internet Society, February 1990.

[9] E. W. Dijkstra, A note on two problems in connexion with graphs. Numerische Mathematik 1: 269–271, 1959.

[10] Floyd, R. W. (1962). Algorithm 97: Shortest path. Communications of the ACM,Volume 5 No 6.

[11] Bellman, Richard, "On a routing problem", Quarterly of Applied Mathematics 16: 87–90, 1958

[12] Michal Pióro, Deepankar Medhi, Routing, Flow, and Capacity Design in Communication and Computer Networks, Morgan Kaufmann Publishers, Elsevier, 2004.

[13] Ian Millington. Artificial intelligence for games. Morgan Kaufmann Publishers, 2006 by Elsevier Inc. 2006.

[14] W. K. Chen. Theory of Nets: Flows in Networks. John Wiley & Sons, 1990.

[15] D. Bertsekas and R. Gallager. Data Networks—2nd Edition. Prentice Hall, 1992.

[16] Matthias Hentschel, Daniel Lecking, Bernardo Wagner, Deterministic path planning and navigation for an autonomous forklift truck, Proceedings of IFAC 2007, 2007.

[17] K.T. Vivaldini, J. P. M. Galdames, T. B. Pasqual, R. C. Araújo, R. M. Sobral, M. Becker, and G. A. P. Caurin" Robotic Forklifts for Intelligent Warehouses: Routing, Path Planning, and Autolocalization", in IEEE – International Conference on Industrial Technology, Viña del Mar – Valparaíso, Chile, Mar. 2010.

[18] K.T. Vivaldini, J. P. M. Galdames, T. B. Pasqual, M. Becker, and G. A. P. Caurin, “Intelligent Warehouses: focus on the automatic routing and path planning of robotic forklifts able to work autonomously,” Mechatronics Systems: Intelligent Transportation Vehicles, 2010.

[19] I. Beker, V. Jevtic and D. Dobrilovic, Using shortest-path algorithms for forklift route planning and optimization. XV International scientific conference on industrial systems, pp 285-290, September, 14th-16th Novi Sad, Serbia, 2011.

Algoritmi najkraće putanje kao alati za optimizaciju unutrašnjeg saobraćaja

Ivan Beker, Vesna Jevtić, Dalibor Dobrilović

Primljeno (15 decembar 2011); Recenzirano (24 januar 2012.); Prihvaćeno (08 februar 2012.)

Rezime

Ovaj rad se bavi poslovnim procesima i poboljšanjem produktivnosti u cilju smanjenja troškova bilo koje vrste, posebno eliminisanjem otpada. Stoga, automatizacija logističkih procesa je veoma važna, i u ovom slučaju predstavlja izvor smanjenja jednog od najvećih troškova: unutrašnji saobraćaj. U praksi postoji nekoliko modela za odabir puta i osnovni cilj ovog rada jeste da istraži implementaciju jednog od modela – algoritma najkraće putanje za optimizaciju kretanja viljuškara. Računanjem optimalne putanje za viljuškare, saobraćajni putevi se skraćuju i rad u skladištima se smanjuje. Ovi algoritmi mogu da se primene za druge tipove vozila i za druge tipove skladišta takođe. Pregled metoda optimizacije se koristi za identifikaciju najpogodnije metode.

Ključne reči: *logistički troškovi, putanja viljušaka, optimizacija, Dijkstra algoritam*