Open Platform for Innovation in Logistics - Agent Optimization

Ladislav Körösi
Institute of Robotics and Cybernetics
Slovak University of Technology
Ilkovičova 3, Bratislava 81219, Slovak Republic
ladislav.korosi@stuba.sk

František Duchoň
Institute of Robotics and Cybernetics
Slovak University of Technology
Ilkovičova 3, Bratislava 81219, Slovak Republic
frantisek.duchon@stuba.sk

Pavol Lukáč
Institute of Robotics and Cybernetics
Slovak University of Technology
Ilkovičova 3, Bratislava 81219, Slovak Republic
xlukacp1@stuba.sk

***Abstract*—This paper is devoted to design a new module for the Open Platform for Innovation in Logistics (OPIL) library, which is a part of European Union Horizon 2020 project "Grow your manufacturing business" known under acronym Better Factory (Horizon 2020). The goal of the new module is to calculate the optimal number of agents required for material transport in logistics. This task is highly actual in the early automation of logistics or in later extension stage. The new module is designed as independent Docker image and can be connected with the new connector Material Flow to the Real Time Locating system. Two examples demonstrates the proposed method implemented for Internet of Things (IoT) system.**

***Index Terms*—logistics, material flow, agent, optimization, IoT, Docker**

I. Introduction

Industry 4.0 principles are applied in current modern production systems. These principles assume a strong synergy of deployed systems and their optimization. Material logistics is one of the key systems, especially in highly automated productions.

The supply chain faces mounting challenges like slowdowns, labor shortages, congested ports, and container backlogs. At the same time, the sector is experiencing a wholesale makeover [2], [3]. Manufacturers are embracing digital transformation in the post-pandemic world. It has become a top priority in an environment when the pressures of a broken supply chain continue to shape 2022 [4].

Therefore, for many reasons, digitizing material flows in production is necessary. For this, discrete event simulation and statistical analysis capabilities optimize material handling, logistics, machine utilization, and labor requirements. It allows to check for bottlenecks quickly, validate transported materials, and view resource utilization over time for multiple process alternatives [5], [6].

There are powerful commercial tools such as Tecnomatix from Siemens [7], Visual Components [8], or Delmia [9] to design such solutions. Most of the time, the trial technique is used in these tools. It requires an experienced engineer. Open systems, such as the OPIL library [10] allow the implementation of more advanced functions. At the same time, their use is also suitable for small and medium-sized enterprises. In our research, we decided to use this platform and offer SMEs an automated system that determines the optimal number of logistics agents for given material flows.

978-1-6654-8775-7/22/$31.00 ©2022 IEEE

II. Open Platform for Innovation in Logistics

OPIL is the Open Platform for Innovations in Logistics. This platform is meant to develop value-added services for the logistics sector in small-scale Industry 4.0 contexts such as manufacturing SMEs. It provides an easily deployable suite of applications for the rapid development of complete logistics solutions, including components for task scheduling, path planning, automatic factory layout generation and navigation, and logistics agent optimization [11].

OPIL also provides connectivity to equipment for optimal material handling on the factory floor. These include (but are not limited to) mobile robots, AGVs, forklifts, workers, and sensors, as well as the IT infrastructure of the factory as various warehouse management and ERP (Enterprise Resource Planning) systems [11].

OPIL provides a ready integration to a state-of-the-art 3D factory simulator (with mobile robots, AGVs, forklifts, workers, and sensors) which allows for complete development and virtual testing of the logistics solutions before actual implementation. The 3D simulator provides support for estimating investment and guides the orchestration of the deployment tasks. A messaging system at the heart of OPIL, based on the FIWARE open architecture, handles interactions among the several components of the platform as well as with the external ones. The open architecture of OPIL (Fig. 1) supports the usage of open-source frameworks such as ROS (Robot Operating System) for the development of new components, e.g., perception, mobile manipulation, etc. [11]

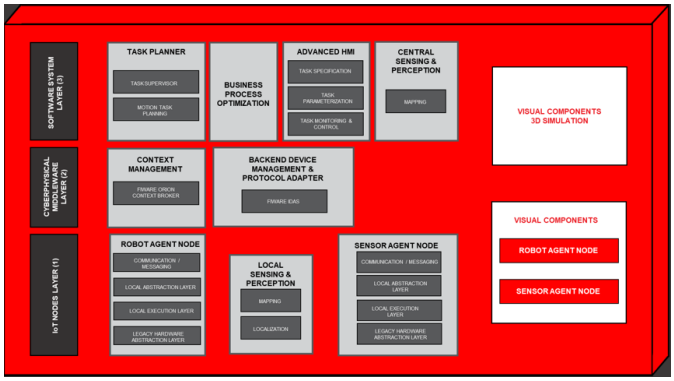

Fig. 1. Architecture of OPIL.

The Logistics Library consists of three layers. In each layer the individual modules and submodules ensure the overall operation of the library.

The first layer IoT Nodes interacts with the physical world. One of the tasks of this layer is to ensure the correct control of robots through the Robot Agent Node module, which not only controls the robots but also acts as an interface between the first and second layer, from which it receives information about trajectories. These received entities are translated into a language understandable by Robot Operating System (ROS) [11].

The Sensor Agent Node module is responsible for connecting and sending data from sensors. The information is sent as an entity in Next Generation Service Interface (NGSI) format via the HTTP REST API developed by FIWARE. Before being sent, the raw data are processed in the Local Execution layer [11].

Communication between individual modules is mediated by the second layer called Cyber Physical Middleware. Specifically by the Context Management module, which is implemented by FIWARE Orion Context Broker (OCB). The module acts as a central information hub, receiving and sending data in NGSI format and then storing them in the MongoDB database [11].

The translation from and to the NGSI format takes place in the Backend Device Management module and the protocol adapter, which is implemented by FIWARE Intelligent Data Advanced Solution [11].

The Advanced Widget mash-up HMI module implements FIWARE Wirecloud, which allows end users without programming skills to easily create web applications and dashboards [11].

The Software System layer consists of three modules. The Task Planner module has the task of redistributing and monitoring executed motion tasks using the Task Supervisor submodule, and the Motion Task Planning submodule is responsible for planning the shortest, fastest routes and ensures communication between robots, AGVs [11].

The Sensing & Perception (SP) module provides information suitable for precise planning of robot movement and the position of robots on the map. The robot has a local SP module that creates its own map, then sends this map to the central SP module on the servers and updates the map with new sensor values. The local map can also be accessed by the Robot Agent Node (RAN) module. SP module also includes the Simultaneous Localization And Mapping function, which can create a map if none was specified [11].

The last part is the HMI module, which includes a web application with its own Mongo DB database. The module serves as a user interface in which the position of the robots their possible routes can be monitored. It allows entering and monitoring of tasks being performed. This web application uses the FIWARE Ngsijs Javascript library to establish a connection between the OCB and the NGSI proxy. HMI also includes the Agent Optimization module, which is used to calculate the optimal number of agents for the transportation of material [11].

III. Agent Optimization

In this section the agent optimization methodology and implementation will be described.

A. Methodology

The Agent Optimization is a newly developed module in the Logistics library. It's based on a method published in [1]. The module can be used in determining equipment requirements as how many AGV's, forklift trucks, humans, etc. will be required to satisfy a specific flow rate in logistics. This method is focused on a vehicle-based approach, where the agent can be anything that transports material. The agent parameters which are used in the optimization are

- v_c - agent speed [m/min],
- T_L - time to load the agent at load station [min],
- T_U - time to unload the agent at unload station [min],
- c - agent capacity [pcs],
- A - availability of the agent,
- F_t - traffic factor,
- E_w - worker efficiency.

The agent speed is given by the vehicle parameters. The average load and unload times are determined by measuring the duration of the operations. The capacity is the maximum number of objects that can be delivered by the agent. The availability is a reliability factor defined as the proportion of total shift time that the agent is operational. Traffic factor is defined as a parameter for estimating the effect of losses due to traffic congestion. If there is no blocking of agents (for example at intersections), waiting in queues at load and unload stations, the factor is chosen $F_t = 1.0$. The logistics efficiency depends not only on traffic factor, but also on worker efficiency who drives the trucks. If the agent is AGV, the factor is chosen $E_w = 1.0$.

Beside the described agent parameters the distance and flow rate matrices are needed. They are NxN matrices, where the rows represent the source stations and the columns the destination stations. In the distance matrix, the number $d = 10$ at row 2 and column 3 defines the distance from station 2 to station 3 in meters. In the flow rate matrix, the number

$f = 100$ at row 2 and column 3 defines the material flow rate from station 2 to station 3 in pieces. In the proposed algorithm negative numbers (default -1) indicate, that the agent is returning from current station to the load station.

Distance that agent travels between load and unload station is given by

$$\begin{aligned} F_{ij} \geq 0, L_d &= \frac{\frac{\sum_{i,j=1}^{N} F_{ij} * D_{ij}}{c}}{\sum_{i,j=1}^{N} F_{ij}} \\ F_{ij} < 0, L_e &= \sum_{i,j=1}^{N} D_{ij} \end{aligned} \tag{1}$$

where c is the agent capacity, F is the flow rate matrix, D is the distance matrix, i and j are elements of matrices where i represents the rows (start stations) and j the columns (destination stations) and L_e is the distance that the agent travels empty until the start of the next delivery cycle.

The total cycle time per delivery per agent is given by

$$T_C = T_L + \frac{L_d}{v_c} + T_U + \frac{L_e}{v_c} \tag{2}$$

The number of required deliveries is calculated as

$$F_{ij} \geq 0, w = \sum_{i,j=1}^{N} F_{ij} \tag{3}$$

The ideal cycle time per delivery per agents is given as

$$WL = wT_C \tag{4}$$

Available time per hour per agent is given as

$$AT = 60AF_tE_w \tag{5}$$

where A is the agent availability, F_t is the traffic factor and E_w is the operator efficiency. Finally the optimal number of agents is determined as

$$AN = \frac{WL}{AT} \tag{6}$$

The result is rounded to integer number.

B. Implementation

The Logistics library is based on Docker image solutions. We decided to create an independent Docker image with the presented optimization algorithm. This approach ensures independent use of the new logistics module. The architecture of the proposed module is in Fig. 3.

The Docker image contains the application and a MongoDB database. The database is used to store the agent parameters and matrices that are used in optimization. The database is saved in the host Ubuntu based PC. The interaction with the module can be divided to

- Material Flow - It's a new connector in Logistics library between Real Time Locating System (RTLS). The aim of the connector is provide inputs for Agent Optimization collected from databases used by RTLS.
- Manual entry - entering the data manually into MongoDB.
  - Using MongoDB commands.
  - Using the new Human Machine Interface (Fig. 2.).
- Other application - using your own application to access the MongoDB through default port and start the optimization.

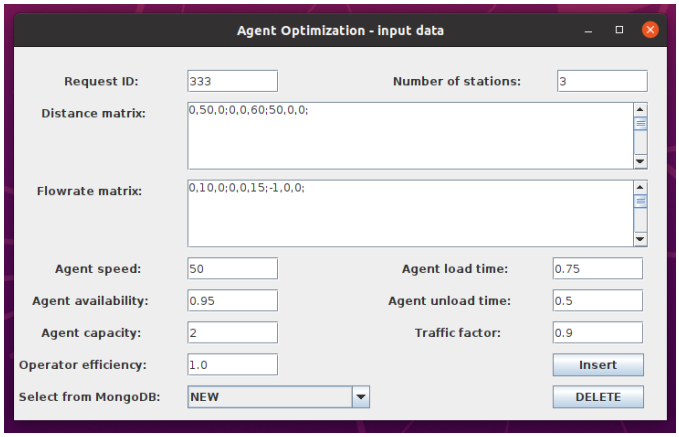

Fig. 2. Human machine interface for Agent Optimization.

An example of command to store a document in MongoDB for Agent Optimization is **db.AGENTS.insertOne("request_id" : "123", "N" : "3", "distance_matrix" : "0,50,0;0,0,60;50,0,0;", "flowrate_matrix" : "0,10,0;0,0,15;-1,0,0;", "agent_speed" : "50", "agent_load_time" : "0.75", "agent_unload_time" : "0.5", "agent_capacity" : "2", "agent_availability" : "0.95", "traffic_factor" : "0.9", "operator_efficiency" : "1.0", "result_optimal_number_of_agents" : "1.3694", "result_optimal_number_of_agents_rounded" : "2", "result_DONE" : "0"**). Command insertOne inserts one document into DB. All values are stored as strings, because it is easier for the end user to enter them. The request ID is used to identify the entered optimization by the end user. The matrices are encoded as line of characters. The row of the matrix ends with ";". Numbers (elements) in rows are separated with ",". Variables result_optimal_number_of_agents and result_optimal_number_of_agents_rounded are the calculated optimal values. Last variable result_DONE has two states. If its state is 0 and the application is executed, the optimization will be performed and the value will be overwritten to 1. If its state is 1, the optimization will be skipped.

The HMI for the Agent Optimization module was developed in Java. The MongoDB Java Driver library was used to connect the HMI to the MongoDB database. The HMI can insert data directly into the database, as well as to delete or edit them. The input data are checked before entering into database. The validation checks the correct format, size, range, prohibited characters, the occurrence of duplicate input data or IDs. An example of Agent Optimization execution is in Fig. 4.

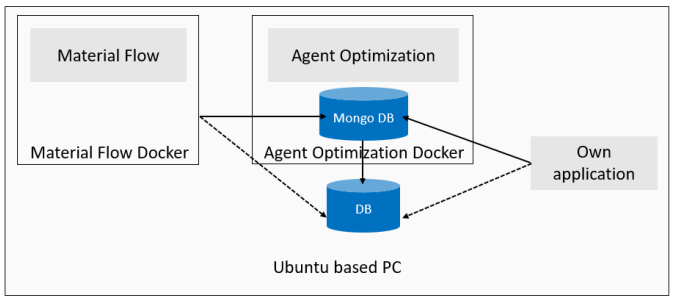

Fig. 3. Architecture of Agent Optimization.

```
ncr@ubuntu:~$ sudo docker exec -it ao-instance1 /code/main
Optimization started...reading request_id (333)
reading N
reading distance_matrix
reading distance_matrix
reading agent_speed
reading agent_load_time
reading agent_unload_time
reading agent_capacity
reading agent_availability
reading traffic_factor
reading operator_efficiency
Ideal cycle time per delivery per vehicle is [min] 2.810
Ideal cycle time per delivery per vehicle is [min / hr] 70.250
Available time per hour per vehicle [min/hr per vehicle] 51.300
Number of required vehicles 1.369
Rounded number of required vehicles 2
result from optim. function 1.369396
```

Fig. 4. Agent Optimization - Example of optimizatiom textual output.

IV. Examples

A. Example 1

Case A

The first example consist from 3 stations (Fig. 5). The agent parameters are: agent speed 50 m/min, time to load 0.75 min, time to unload 0.5 min, agent capacity 1 pcs, agent availability 0.95, traffic factor 0.9 and operator efficiency 1.0. The distance matrix (Tab. I) and flow rate matrix (Tab. II) is defined as The calculated ideal cycle time per delivery per agent is 3.37 min, the ideal cycle time per delivery per agent is 84.25 min/hr, the available time per hour per agent 51.3 min/hr per agent and the number of required agents 1.642 which is rounded to 2.

TABLE I
Distance matrix

	Station 1	Station 2	Station 3
Station 1	0	50	0
Station 2	0	0	60
Station 3	50	0	0

TABLE II
Flow rate matrix

	Station 1	Station 2	Station 3
Station 1	0	10	0
Station 2	0	0	15
Station 3	-1	0	0

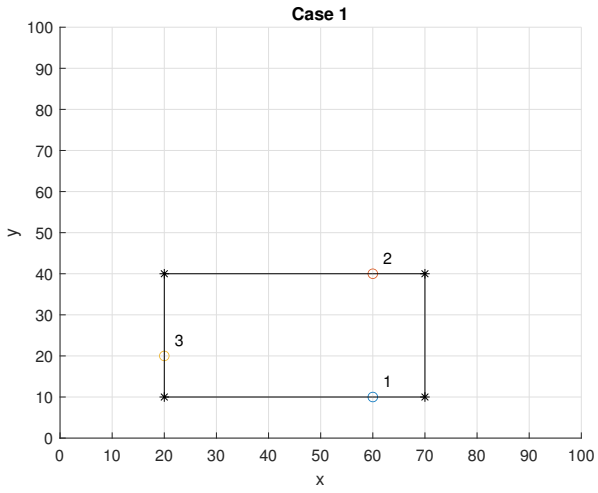

Fig. 5. First example.

Case B

In the second case we left the same distance and flow matrices with agent parameters except the agent capacity which was increased to 2.

The calculated ideal cycle time per delivery per agent is 2.81 min, the ideal cycle time per delivery per agent is 70.25 min/hr, the available time per hour per agent 51.3 min/hr per agent and the number of required agents 1.369 which is rounded to 2. We can see that the cycle times were reduced by increasing the agent capacity, which has slightly reduced the required agent number.

B. Example 2

Case A

The second example consist from 7 stations (Fig. 6). The agent parameters are: agent speed 50 m/min, time to load 0.75 min, time to unload 0.5 min, agent capacity 1 pcs, agent availability 0.95, traffic factor 0.9 and operator efficiency 1.0. The distance matrix (Tab. III) and flow rate matrix (Tab. IV) is defined as

TABLE III
Distance matrix

	St. 1	St. 2	St. 3	St. 4	St. 5	St. 6	St. 7
St. 1	0	65	0	0	0	0	0
St. 2	0	0	45	0	0	0	0
St. 3	0	0	0	50	0	140	0
St. 4	0	0	0	0	70	0	0
St. 5	0	0	0	0	0	120	0
St. 6	0	0	0	0	0	0	65
St. 7	45	0	0	0	0	0	0

The calculated ideal cycle time per delivery per agent is 3.61 min, the ideal cycle time per delivery per agent is 3610.0

TABLE IV
FLOW RATE MATRIX

	St. 1	St. 2	St. 3	St. 4	St. 5	St. 6	St. 7
St. 1	0	200	0	0	0	0	0
St. 2	0	0	200	0	0	0	0
St. 3	0	0	0	100	0	100	0
St. 4	0	0	0	0	100	0	0
St. 5	0	0	0	0	0	100	0
St. 6	0	0	0	0	0	0	200
St. 7	-1	0	0	0	0	0	0

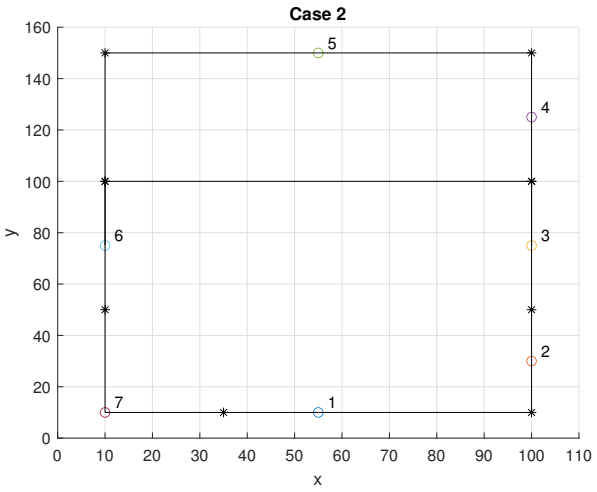

Fig. 6. Second example.

min/hr, the available time per hour per agent 51.3 min/hr per agent and the number of required agents 70.37 which is rounded to 71.

Case B

In the second case we left the same distance and flow matrices with agent parameters except the agent capacity which was increased to 100.

The calculated ideal cycle time per delivery per agent is 2.165 min, the ideal cycle time per delivery per agent is 2164.6 min/hr, the available time per hour per agent 51.3 min/hr per agent and the number of required agents 42.195 which is rounded to 43.

V. CONCLUSION

This paper presented a new module of the Logistics Library to determine the optimal number of required agents in logistics for material transport. The module was developed during the Grow your manufacturing business project known as Better Factory (BF), which helps small and medium manufacturers to redesign their product portfolio, increase and optimize their logistics and production. The module was verified using simple simulation examples and will be tested in the next BF Knowledge Transfer Experiments (KTEs). In each example two cases were presented. In case A, the agent had less capacity as in case B, which should intuitively in lead to higher number of required agents. This was proven by the optimization results.

ACKNOWLEDGMENT

This work has been supported by Grant VEGA 1/0049/20 of the Slovak Scientific Grant Agency and by Grow your manufacturing business - Better Factory, Grant agreement ID 951813, DOI 10.3030/951813 and by Grant VEGA 1/0049/20 of the Slovak Scientific Grant Agency.

REFERENCES

[1] M. Groover, “Automation, production systems, and computer-integrated manufacturing“ Pearson, 4th ed., January 5, 2015, pp. 816, ISBN-13 978-0133499612.

[2] A. Dekhne, G. Hastings, J. Murnane, F. Neuhaus, “Automation in logistics: Big opportunity, bigger uncertainty“. McKinsey, 2019, pp. 1–12.

[3] B. Nitsche, F. Straube, M. Wirth, “Application areas and antecedents of automation in logistics and supply chain management: a conceptual framework“ In Supply Chain Forum: An International Journal, 2021, vol. 22, no. 3, pp. 223–239, Taylor & Francis.

[4] “Top 5 ways to optimize logistics management“, stefanini.com, https://stefanini.com/en/trends/news/top-5-ways-to-optimize-logistics-management (accessed: July 8, 2022).

[5] “Optimize production logistics & material flow - Simulate logistics and material flow for improved system performance“, www.plm.automation.siemens.com, https://www.plm.automation.siemens.com/global/en/products/tecnomatix/logistics-material-flow-simulation.html (accessed: July 8, 2022).

[6] M. Pekarcikova, P. Trebuna, M. Kliment, M. Dic, “Solution of bottlenecks in the logistics flow by applying the kanban module in the tecnomatix plant simulation software“, Sustainability, 2021, vol. 13, no. 14, 7989, https://doi.org/10.3390/su13147989.

[7] P. A. Russkikh, D. V. Kapulin, “Simulation modeling for optimal production planning using Tecnomatix software“ In Journal of Physics: Conference Series, 2020, vol. 1661, no. 1, p. 012188, IOP Publishing.

[8] A. Laemmle, S. Gust, “Automatic layout generation of robotic production cells in a 3D manufacturing simulation environment“ Proced, 2019.

[9] H. Zhong, X. Zhang, J. Hu, S. Liu, X. Shao, “Productivity analysis and optimization of aircraft assembly line based on Delmia-Quest“, In Asia-Pacific International Symposium on Aerospace Technology, Singapore, Springer, 2018, pp. 1150–1159.

[10] M. Seder, L. Petrović, J. Peršić, G. Popović, T. Petković, A. Šelek, B. Bićanić, I. Cvišić, D. Josić, I. Marković, I. Petrović, A. Muhammad, “Open platform based mobile robot control for automation in manufacturing logistics“, IFAC-PapersOnLine, 2019, vol. 52, no.22, pp. 95–100.

[11] “Open platform for innovations in logistcs“, readthedocs.io, https://opil-documentation.readthedocs.io/en/latest/ (accessed: July 8, 2022).