Task Allocation Algorithms for Mobile Robots, Application to Industrial Logistics

Arnaud Belhomme, François Guerin

La robotisation dans les entrepôts logistiques

PÔLE INGÉNIEUR
LOGISTIQUE
LE HAVRE NORMANDIE

CAHIERS DE LA LOGISTIQUE

ISSN : 2778-0317

CAHIERS DE LA LOGISTIQUE

CL2024 – N° 05

La robotisation dans les entrepôts logistiques

Décembre, 2024

TASK ALLOCATION ALGORITHMS FOR MOBILE ROBOTS, APPLICATION TO INDUSTRIAL LOGISTICS

Arnaud BELHOMME[1], **François GUERIN**[2]

[1]GREAH, University Le Havre Normandie
Le Havre, France
arnaud.belhomme@univ-lehavre.fr

[2]GREAH, University Le Havre Normandie
Le Havre, France
francois.guerin@univ-lehavre.fr

RESUME

Dans cette publication, il est exploré les algorithmes d'allocation de tâches pour les robots mobiles appliqués à la logistique industrielle. Nous y examinons les systèmes de décision centralisés et décentralisés, selon leur flexibilité, robustesse et évolutivité. L'étude se concentre sur l'optimisation de l'allocation des tâches dans un système multi-robots en utilisant diverses stratégies telles que la programmation linéaire, le calcul itératif, la résolution de conflits par consensus, les modèles d'apprentissage automatique et l'apprentissage par renforcement. Les méthodes proposées sont évaluées par des simulations dans un environnement industriel, notamment au sein d'une entreprise d'assemblage de nacelles de moteurs d'avion. L'objectif est d'améliorer l'efficacité en permettant aux robots d'assigner des tâches de manière autonome en fonction de leurs capacités et contraintes. La recherche se conclut par une comparaison des performances des algorithmes, offrant des perspectives pour de futures applications dans l'automatisation industrielle.

MOTS CLÉS: Allocation de tâches, Robots mobiles, Logistique industrielle.

ABSTRACT

This publication explores task allocation algorithms for mobile robots in industrial logistics. It examines both centralised and decentralised control methods, according to their flexibility, robustness, and scalability. The study focuses on optimising task allocation in a multi-robot system using different strategies such as linear programming, iterative calculation, conflict resolution by consensus, machine learning models, and reinforcement learning. The proposed methods are evaluated through simulations in an industrial setting, specifically within an aircraft engine nacelle assembly company. The goal is to enhance efficiency by allowing robots to autonomously assign tasks based on their capabilities and constraints. The research concludes with a comparison of the algorithms' performance, providing insights for future applications in industrial automation.

KEYWORDS: Task Allocation, Mobile Robots, Industrial Logistics.

1. Introduction

In recent years, industrial enterprises have constantly sought the development of their products, processes and organisation, with the aim of maintaining a competitive advantage. In this context, industrial logistics provides essential support which makes it possible to optimise the supply of tools and raw materials to production lines, Shang (2022). Mobile robots form a large and important part of the logistics transportation systems in today's industry are widely used. Mobile robots offered by global manufacturers almost all operate under some form of centralised control where a single central controller coordinates the entire fleet of mobile robots. There is a trend towards decentralised control of these systems where mobile robots make individual decisions that promote the flexibility, robustness and scalability of transport, De Ryck et al. (2021).

Task allocation can be done in both a centralised or decentralised way. In many situations, due to the lack of infrastructure, higher possibility of single-point-of-failure, communication bottlenecks, loss of communication with the central node, etc., centralised task allocation becomes impractical. Especially the applications like war-field, disastrous scenario, underwater exploration where any infrastructure is not available, centralised allocation of tasks is not feasible, Mahato et al. (2023). Multi-agent systems are decentralised control systems where the autonomous agents in the system coordinate and cooperate with each other in order to achieve a global goal, Teck et al. (2023). In recent years, there has been a growing association between robotics and artificial intelligence, aiming to enable robots to make autonomous decisions, Sharma et al. (2023).

In a traditional decentralised task allocation strategy, first, a leader is elected, which oversees the overall task allocation. Such strategies will also have similar problems as a centralised solution in hostile scenarios. It may ultimately lead to the frequent election of the leaders, wasting time and energy. Decentralised task allocation without the intervention of a dedicated leader has also been considered in methods following various bidding-based schemes. Given a set of tasks, a robot can derive a bid value indicating its suitability in carrying out each of the tasks. Optimal distributed allocation of the tasks can be ensured using a simple two-step algorithm. In the first step, the robots would exchange the bid-values for each task with each of the robots. Next, each robot can run the task allocation algorithm locally as all of them possess the same input (bid value). This algorithm involves broadcast data sharing, which is a very costly operation in terms of the amount of communication needed in the underlying network, Mahato et al. (2023).

Multi-Robot Task Allocation (MRTA) can be solved by various optimisation approaches such as Integer Linear Program (ILP), heuristics, and meta-heuristics algorithms, Chakraa et al. (2023). Hungarian algorithm: A decentralised task allocation algorithm based on the Hungarian approach is proposed in Ismail and Sun (2017). The proposed algorithm guarantees an optimal solution as long as the agent network is connected, i.e., the second smallest eigenvalue of the Laplacian matrix of the agent graph is greater than zero. Patel et al. (2020) propose an approach for decentralised task allocation based on a decentralised Genetic Algorithm (GA). The approach parallelizes a genetic algorithm across the team of agents, making efficient use of their computational resources. Buckman (2018) presents a fully decentralised, dynamic task allocation algorithm that extends the Consensus-Based Bundle Algorithm (CBBA) to allow for allocating new tasks. Nayak et al. (2020) compare the performance of five state of the art decentralised task allocation algorithms under imperfect communication conditions. The decentralised algorithms we consider are CBAA (Consensus Based Bundle Algorithm), ACBBA (Asynchronous Consensus Based Bundle Algorithm), DHBA (Decentralised Hungarian Based Algorithm), HIPC (Hybrid Information and

Plan Consensus) and PI (Performance Impact). A decentralised allocation algorithm for distributed supply chains with critical tasks: Binetti et al. (2013) considers the problem of allocating tasks to a network of interconnected nodes in a supply chain, considering functional heterogeneity, resource constraints, and critical tasks whose assignment has to be considered mandatory. The proposed approach is an auction-based algorithm which uses a consensus algorithm to obtain a conflict-free solution fulfilling all the constraints. Numerical simulations and a comparison with a centralised optimisation algorithm are performed to evaluate the effectiveness of the proposed approach. With recent advances in mobile robotics, autonomous systems, and artificial intelligence, there is a growing expectation that robots are able to solve complex problems. Many of these problems require multiple robots working cooperatively in a multi-robot system. Complex tasks may also include the interconnection of task-level specifications with robot motion-level constraints. Many recent works in the literature use multiple mobile robots to solve these complex tasks by integrating task and motion planning, Antonyshyn et al. (2023).

In our work, we propose to compare different families of task allocation algorithms, with the aim of identifying the most relevant ones which will be applied to an example of an industrial case.

2. Problem description

We consider a fleet of R mobile robots, with N tasks to complete. Each robot is endowed with unique abilities that are relevant to the completion of these tasks. All robots can complete the tasks. The objective is to develop an embedded algorithm for each robot allowing it to assign itself to the task for which it is most competent, due to battery level (state of charge) and time (distance, speed). Then provide a consensus mechanism in the event of a task assignment conflict between several robots.

2.1 Industrial Application Example

The solution proposed in this paper will be implemented in an aircraft engine nacelle assembly company, as part of a line-side supply automation project.

Consider the example in figure 1 with 3 robots and 5 workstations. Each operator has a replenishment request push button for his workstation. The robot fleet is then responsible for managing the replenishment of component bins at various workstations. The supply of full bins (using a First In First Out table) is carried out by an operator, as well as the evacuation of empty bins.

For the purpose of a modelization (figure 2), let us consider the following notations :

- "N" operator workstations ($N \geq 1$), with 2 parts:
  - 1 : full bins => capacity = $Co_F(1)$
  - 2 : empty bins => capacity = $Co_E(1)$
- "R" mobile robots ($R \geq 1$)
- S_F : full bin station => capacity = $C_F(25)$
- S_E : empty bin station => capacity = $C_E(25)$

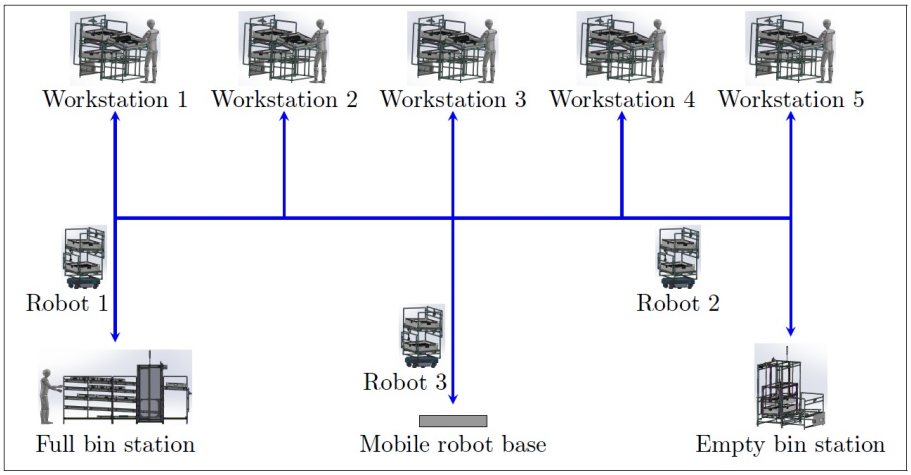


Figure 1: Industrial application (example for 3 robots and 5 workstations).

- O_{T_1}, ... O_{T_N} : workstation operators
- O_F : full bin station operator
- O_E : empty bin station operator
- Each workstation and station is equipped with a "level sensor" connected to an IoT network.
- Each mobile robot can transport "B_{R_1}, ... B_{R_N}" bins and perform actions:
  - GoS_F = go to full bin station + bin recovery
  - GoO_{F_1}, ... GoO_{F_N} = go to the "full" operator workstation 1, ... N + bin deposit
  - GoO_{E_1}, ... GoO_{E_N} = go to the "empty" operator workstation 1, ... N + bin recovery
  - GoS_E = go to empty bin station + bin deposit
  - $GoBase$ = return to base (battery charge)
- The navigation function of mobile robots is supposed to be perfect.

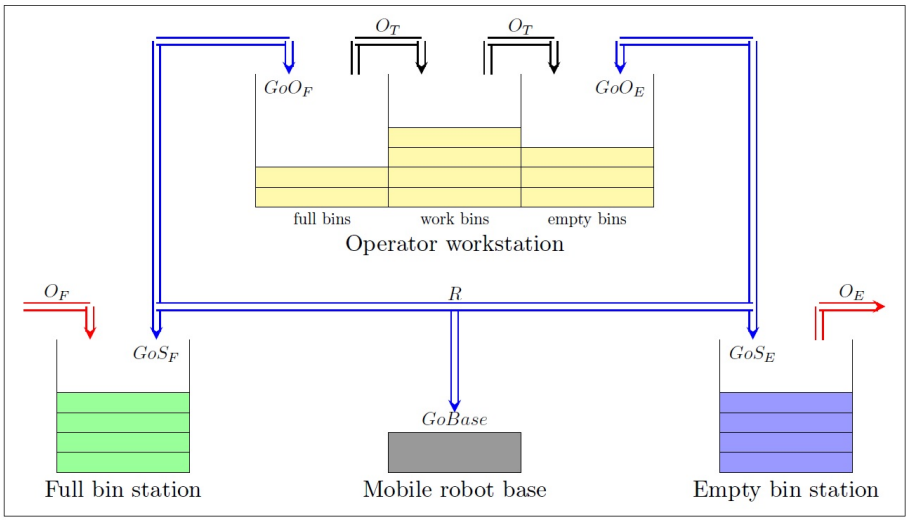


Figure 2: Workstation modelization (one workstation represented).

3. Contribution

3.1 Formulation of the Problem

Consider a set of robots $R = \{r_1, \ldots r_n\}$ and a set of tasks $T = \{t_1, \ldots t_m\}$. Each robot r_i has attributes including position p_i, battery level b_i, speed s_i, and payload capacity c_i.

3.2 Algorithms developed

We chose to implement different task allocation algorithms in order to compare them, both in terms of efficiency, but also in calculation time. Efficiency is measured by calculating the sum of the task capacities assigned to the different robots. The sum of the greatest capacity then corresponds to the best overall task allocation efficiency.

The algorithm 1 (Task Allocation with Mixed-Integer Linear Programming - MILP) uses linear programming to maximise task allocation capacities by assigning each robot exactly one task and each task to exactly one robot, subject to the constraints of unique allocation and binary decision variables.

The algorithm 2 (Task Allocation with Iterative Calculation) iteratively assigns tasks to robots using random permutations to maximize the sum of capacities, updating the best allocation found whenever a higher sum of capacities is achieved.

The algorithm 3 (Task Allocation with Conflict Resolution) initially assigns tasks to robots by selecting the task with the maximum capacity for each robot, checks for assignment conflicts, and if conflicts exist, iteratively solves them by randomly permuting task assignments to maximise the sum of capacities.

The algorithm 4 (Task Allocation with Conflict Resolution and Consensus) assigns tasks to robots by selecting the maximum capacity task for each robot, checks for assignment conflicts, and if conflicts exist, uses a local consensus method to resolve them by selecting the robot with the lowest capacity in conflicts or randomly if tied, and marking its task allocation for reassignment.

The algorithm 5 (Task Allocation Based on Ranking) ranks robot-task pairs based on a custom scoring criterion, sorts these pairs by score, and then allocates tasks to robots sequentially based on this ranking, ensuring each robot and task is assigned only once.

The algorithm 6 (Task Allocation based on a Linear Regression Machine Learning Model) imports a dataset from an Excel file, splits it into training and testing sets, trains a linear regression model on the training data, evaluates the model on the test data, and then uses the trained model to predict outputs based on a given input matrix.

The algorithm 7 (Task Allocation based on Machine Learning with Perceptron Model) iteratively trains a Perceptron model for each robot using a dataset imported from an Excel file, evaluates the model's performance, and uses it to predict task allocations based on given capacity data, storing the predictions in a task allocation matrix.

The algorithm 8 (Task Allocation based on a Reinforcement Learning Model with a Q-Learning Method) employs Q-learning with an epsilon-greedy strategy over multiple episodes to optimise task allocation for robots, updating a Q-matrix based on the total capacity achieved in each episode and ultimately determining the best task assignment for each robot.

Algorithm 1 Task Allocation with Mixed-Integer Linear Programming (MILP)

INITIALISATION:
Import linear programming solver
Create the objective coefficients (f) to maximise the sum of capacities
Initialise integer variables for task allocation (intcon)
Initialise equality constraints matrix (Aeq) with zeros
Initialise equality constraints bounds (beq) with ones for unique allocation constraint
CALCULATION:
for each robot i **do**
    Set sum constraints in Aeq for each robot
end for
for each task j **do**
    Set sum constraints in Aeq for each task
end for
Set bounds for each variable representing task allocation to a robot (0 or 1)
Solve the MILP problem with linprog
OUTPUTS:
Reshape the result to match the number of robots and tasks
for each robot i **do**
    Print the allocated task for the robot
end for

Algorithm 2 Task Allocation with Iterative Calculation

INITIALISATION:
Initialise number of iterations
Initialise TaskAllocation matrix
CALCULATION:
for each iteration **do**
    Randomly permute the tasks and reshape to (numTasks, 1) for TaskAllocation
    Set NewCapaSum to 0
    for each robot i **do**
        Add the capacity of robot i for its allocated task to NewCapaSum
    end for
    if NewCapaSum is greater than CapaSum **then**
        Copy TaskAllocation to NewTaskAllocation
        Set CapaSum to NewCapaSum
    end if
end for
OUTPUTS:
for each robot i **do**
    Print the allocated task for the robot (R{$i + 1$} -> T{int(NewTaskAllocation[i]) + 1})
end for

Algorithm 3 Task Allocation with Conflict Resolution

INITIALISATION:
Initialise number of iterations
Initialise TaskAllocation matrix
CALCULATION:
for each robot i **do**
    Find the maximum value in the capacity of robot i
    Find the positions of the maximum value
    if more than one maximum value (conflict) **then**
        Randomly choose one of the maximum positions
    else
        Directly assign the task
    end if
end for
if all tasks are uniquely allocated **then**
    Copy TaskAllocation to NewTaskAllocation
else
    for each iteration **do**
        Randomly permute the tasks and reshape to (numTasks, 1) for TaskAllocation
        Set NewCapaSum to 0
        for each robot i **do**
            Add the capacity of robot i for its allocated task to NewCapaSum
        end for
        if NewCapaSum is greater than CapaSum **then**
            Copy TaskAllocation to NewTaskAllocation
            Set CapaSum to NewCapaSum
        end if
    end for
end if
OUTPUTS:
for each robot i **do**
    Print the allocated task for the robot (R{$i + 1$} -> T{int(NewTaskAllocation[i]) + 1})
end for

Algorithm 4 Task Allocation with Conflict Resolution and Consensus

INITIALISATION:
Initialise number of iterations
Initialise TaskAllocation matrix
CALCULATION:
for each robot i **do**
  Find the maximum value in the capacity of robot i
  Find the positions of the maximum value
  if more than one maximum value (conflict) **then**
    Randomly choose one of the maximum positions
  else
    Directly assign the task
  end if
end for
if all tasks are uniquely allocated **then**
  Directly assign the task
else
  for each robot i **do**
    communicate capability $K_i(t_j)$ for each task t_j to other robots for contested tasks
    the robot with the highest $K_i(t_j)$ is assigned the task.
    in case of equal capabilities, the robot that communicated first is assigned the task
  end for
end if
OUTPUTS:
for each robot i **do**
  Set NewTaskAllocation[i] to consensus[i, 0]
  if NewTaskAllocation[i] == -1 **then**
    Print the robot has no task (R{$i+1$} ->)
  else
    Print the allocated task for the robot (R{$i+1$} -> T{int(NewTaskAllocation[i]) + 1})
  end if
end for

Algorithm 5 Task Allocation Based on Ranking

INITIALISATION:
Initialise TaskAllocationFG matrix with zeros of size (numRobots, numTasks)
Initialise an empty list for ranking (classement)
CALCULATION:
for each robot i **do**
  for each task j **do**
    Calculate score as sum of capacities for robot i and task j, adjusted by capacity of robot i for task j
    Append $(i, j, score)$ to ranking list (classement)
  end for
end for
Sort ranking list (classement) by score in ascending order
Initialise allocated_tasks with zeros of size (numTasks)
Initialise allocated_robots with zeros of size (numRobots)
for each $(robot, task, score)$ in ranking list (classement) **do**
  if allocated_robots[robot] == 0 and allocated_tasks[task] == 0 **then**
    Allocate task to robot in TaskAllocationFG
    Mark robot as allocated in allocated_robots
    Mark task as allocated in allocated_tasks
    if sum of allocated tasks equals numTasks **then**
      Break
    end if
  end if
end for
OUTPUTS:
for each robot i **do**
  Print the allocated task for the robot (R{$i+1$} -> T{task number allocated to robot i in TaskAllocationFG})
end for

Algorithm 6 Task Allocation based on a Linear Regression Machine Learning Model

INITIALISATION:
Import train_test_split and LinearRegression from sklearn
Set nbSet
CALCULATION:
Read the first columns from 'DataSet.xlsx' into X_data with nbSet rows
Convert X_data to a numpy array
Read from 'DataSet.xlsx' into y_data with nbSet rows
Convert y_data to a numpy array
Split X_data and y_data into X_train, X_test, y_train, and y_test with 80% training and 20% testing data
Initialise model as LinearRegression
Fit model with X_train and y_train
Calculate score of model on X_test and y_test
Reshape capacity to (1, 25) and store in capacities
Predict y_pred from capacities using model
Round y_pred to the nearest integers
Convert y_pred to integer type
OUTPUTS:
for each robot i from 0 to numRobots - 1 **do**
  Print "R{$i+1$} -> T{y_pred[0][i]}"
end for

Algorithm 7 Task Allocation based on a Machine Learning Model with Perceptron

INITIALISATION:
Import numpy as np
Import train_test_split and Perceptron from sklearn
Initialise TaskAllocationML2 matrix with zeros of size (numRobots, 1)
Set nbSet to 1000
CALCULATION:
for each robot i from 1 to numRobots **do**
  Read the first columns from 'DataSet.xlsx' into X_data with nbSet rows
  Convert X_data to a numpy array
  Read columns from 'DataSet.xlsx' into y_data with nbSet rows
  Convert y_data to a numpy array and flatten it
  Split X_data and y_data into X_train, X_test, y_train, and y_test with 80% training and 20% testing data
  Initialise model as Perceptron with max_iter=40, eta0=0.1, random_state=0
  Fit model with X_train and y_train
  Calculate score of model on X_test and y_test
  Reshape capacity to and store in capacities
  Predict y_pred from capacities using model
  Round y_pred to the nearest integers
  Convert y_pred to integer type
  Store y_pred in TaskAllocationML2 at index $i-1$
end for
OUTPUTS:
for each robot i from 0 to numRobots - 1 **do**
  Print "R{$i+1$} -> T{int(TaskAllocationML2[i])}"
end for

Algorithm 8 Task Allocation based on a Reinforcement Learning Model with a Q-Learning Method

INITIALISATION:
Set num_episodes
Set learning_rate
Set discount_factor
Set exploration_prob
CALCULATION:
Function to choose an action (task) with epsilon-greedy
choose_action(robot)
if random number < exploration_prob **then**
  Return a random task from numTasks
else
  Return the task with the highest Q value for the given robot
end if
# Simulation of episodes
for each episode from 0 to num_episodes - 1 **do**
  Initialise total_capacity to 0
  Assign tasks randomly to task_assignment
  for each robot from 0 to numRobots - 1 **do**
    Set action to the assigned task for the robot
    Get capacity2 from robot_capacities for the robot and action
    Add capacity2 to total_capacity
    Set reward to total_capacity
    Update Q value for the robot and action using the formula:
    $Q[robot, action] = Q[robot, action] + learning\_rate * (reward - Q[robot, action])$
  end for
end for
# Find the best task assignment for each robot
Set best_task_assignment to the task with the highest Q value for each robot
OUTPUTS:
for each robot i from 0 to numRobots - 1 **do**
  Print "R{$i+1$} -> T{int(best_task_assignment[i] + 1)}"
end for

3.3 Comparison and results

In order to compare the different algorithms, we tested them using a set of capacities determined randomly (for each of the tasks carried out by each robot). We then repeated the test on a large number of iterations to identify a trend and make a reliable comparison of the different algorithms.

The algorithm 9 shows how we conducted this comparison. Table 1 summarizes the average task capacities and processing times for five robots and tasks after 1000 iterations. Figure 3 shows the average capacities (y-axis) for each algorithm (x-axis), while Figure 4 depicts their average execution times (y-axis). Both highlight performance differences under the 1000-iteration stopping criterion.

We conclude that Algorithm 8 is the most efficient based on the criterion of the sum of capacities, while Algorithm 5 excels in processing time efficiency. However, for a decentralised approach, Algorithm 4 is the most efficient one, in both sum of capacities andvrunning time performance.

Algorithm 9 Robot Task Allocation

INITIALISATION:
$numRobots \leftarrow 5$
$numTasks \leftarrow 5$
$nbitmax \leftarrow 1000$
$output \leftarrow \text{zeros}(nbitmax, 8 \times 7)$
CALCULATION:
for $nbit \leftarrow 0$ **to** $nbitmax - 1$ **do**
  $capacity \leftarrow$ random between 0 and 1 for each robot-task pair
  Display the capacity table
  print "Robot Abilities:"
  print T
  $start \leftarrow$ current time
  for $algo \leftarrow 1$ **to** 8 **do**
    Run algorithm
    $duration \leftarrow$ round(current time - start, 3)
    Write sum of capacities duration to output matrix
  end for
end for
OUTPUTS:
print $output$

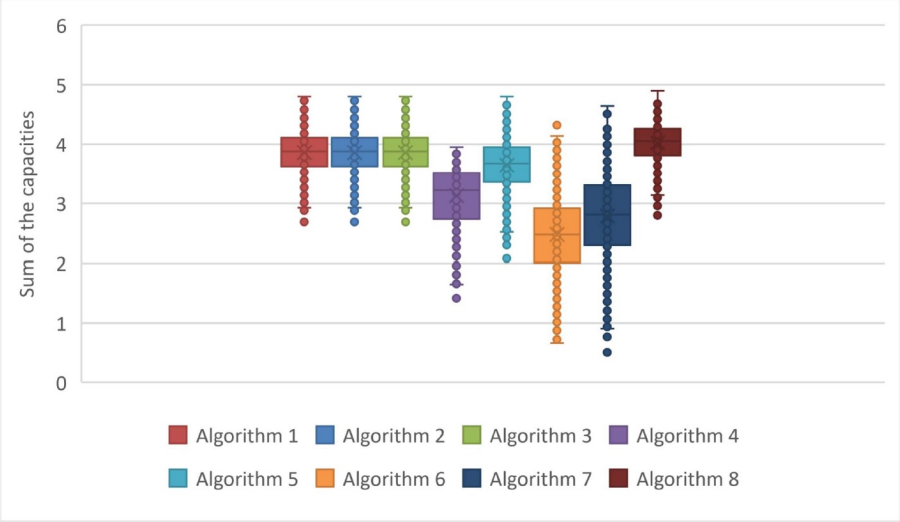


Figure 3: Results of the comparison of algorithms according to calculation of the sum of capacities.

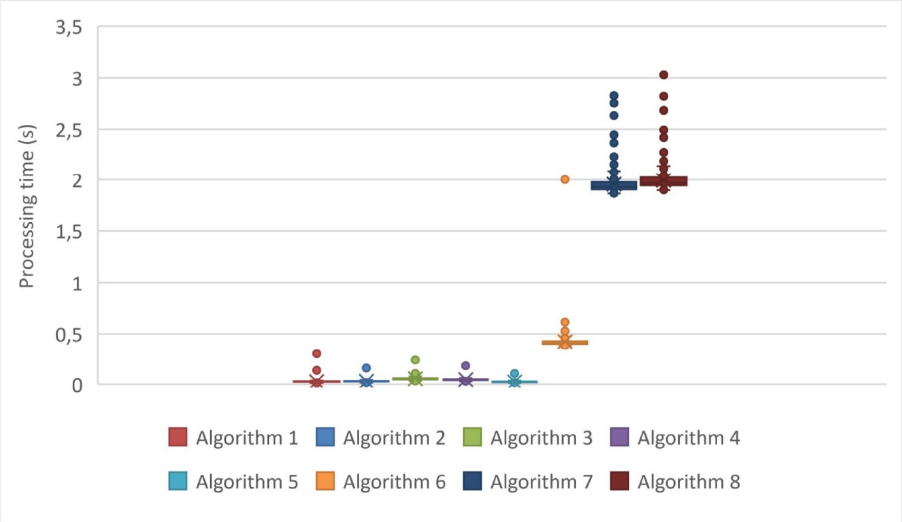


Figure 4: Results of the comparison of algorithms according to execution times.

	Control method	Average of the sums of the capacities	Average processing time (s)
Algorithm 1	centralised	3,86	0,026
Algorithm 2	centralised	3,86	0,029
Algorithm 3	decentralised & centralised	3,86	0,053
Algorithm 4	decentralised	3,13	0,045
Algorithm 5	decentralised & centralised	3,65	0,024
Algorithm 6	centralised	2,48	0,423
Algorithm 7	centralised	2,79	1,954
Algorithm 8	centralised	4,02	1,995

Table 1: Results of comparing algorithms.

3.4 Experimental results

To evaluate our chosen task allocation algorithm, we employed three ROS-based mobile robots. We replicated the industrial application scenario in a laboratory setting, constructing a model within a room equipped with replicas of various workstations. Each robot was loaded with the same program, enabling decentralised operation. All results can be found in Belhomme and Guerin (2024) publication.

4. Conclusions and perspectives

By evaluating various strategies—including linear programming, iterative calculation, conflict resolution, machine learning models, and reinforcement learning—the study identifies the advantages and challenges of each method. The proposed algorithms were tested in a simulated industrial environment, specifically within an aircraft engine nacelle assembly company, demonstrating their potential to enhance operational efficiency. By allowing robots to autonomously assign tasks based on their capabilities and constraints, the research provides valuable insights for improving task allocation processes in industrial automation. The comparative analysis of algorithm performance offers a foundation for future developments in this field, aiming to increase flexibility, robustness, and scalability in industrial logistics systems. The findings underscore the importance of adopting advanced control strategies to optimise multi-robot systems in dynamic and complex industrial settings.

Future research could focus on integrating advanced machine learning techniques, robust communication protocols, and hybrid control systems to enhance task allocation for mobile robots in diverse industrial settings, emphasising energy efficiency, human-robot collaboration, and real-world implementation.

Acknowledgement

We would like to thank the company Safran Nacelles, located in Gonfreville L'Orcher France, for trusting us with the implementation of this project.

References

Antonyshyn, L., Silveira, J., Givigi, S., and Marshall, J. (2023). Multiple Mobile Robot Task and Motion Planning: A Survey. *ACM Computing Surveys*, 55(10):213:1–213:35.

Belhomme, A. and Guerin, F. (2024). Decentralised Bio-inspired Task Allocation for Mobile Robots, Application to Industrial Logistics. In *10th International Conference (IEEE-IFAC) on Control, Decision and Information Technologies (CoDIT 2024) July 01-04, 2024*, Valetta (Malta).

Binetti, G., Naso, D., and Turchiano, B. (2013). A decentralized allocation algorithm for distributed supply chains with critical tasks. *IFAC Proceedings Volumes*, 46(9):192–197.

Buckman, N. N. M. . (2018). *Decentralized task allocation for dynamic, time-sensitive tasks*. Thesis, Massachusetts Institute of Technology. Accepted: 2019-02-05T15:17:44Z.

Chakraa, H., Guérin, F., Leclercq, E., and Lefebvre, D. (2023). Optimization techniques for Multi-Robot Task Allocation problems: Review on the state-of-the-art. *Robotics and Autonomous Systems*, 168:104492.

De Ryck, M., Pissoort, D., Holvoet, T., and Demeester, E. (2021). Decentral task allocation for industrial AGV-systems with resource constraints. *Journal of Manufacturing Systems*, 59:310–319.

Ismail, S. and Sun, L. (2017). Decentralized hungarian-based approach for fast and scalable task allocation. pages 23–28.

Mahato, P., Saha, S., Sarkar, C., and Shaghil, M. (2023). Consensus-based fast and energy-efficient multi-robot task allocation. *Robotics and Autonomous Systems*, 159:104270.

Nayak, S., Yeotikar, S., Carrillo, E., Rudnick-Cohen, E., Jaffar, M. K. M., Patel, R., Azarm, S., Herrmann, J. W., Xu, H., and Otte, M. (2020). Experimental Comparison of Decentralized Task Allocation Algorithms Under Imperfect Communication. *IEEE Robotics and Automation Letters*, 5(2):572–579.

Patel, R., Rudnick-Cohen, E., Azarm, S., Otte, M., Xu, H., and Herrmann, J. W. (2020). Decentralized Task Allocation in Multi-Agent Systems Using a Decentralized Genetic Algorithm. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3770–3776. Conference Name: 2020 IEEE International Conference on Robotics and Automation (ICRA) ISBN: 9781728173955 Place: Paris, France Publisher: IEEE.

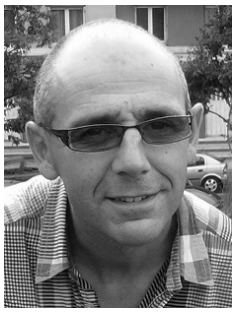
Shang, Y. (2022). Design of an Integrated Approach to Industrial Logistics Information Based on Supply Chain Management. *Advances in Multimedia*, 2022:1–8.

Sharma, N., Pandey, J. K., and Mondal, S. (2023). A Review of Mobile Robots: Applications and Future Prospect. *International Journal of Precision Engineering and Manufacturing*, 24(9):1695–1706.

Teck, S., Vansteenwegen, P., and Dewil, R. (2023). An efficient multi-agent approach to order picking and robot scheduling in a robotic mobile fulfillment system. *Simulation Modelling Practice and Theory*, 127:102789.

Author biography

Arnaud Belhomme began his career in 1996 as a mechanical engineering teacher. In 2014, he expanded his teaching scope to include robotics and operations management at the University of Le Havre, France, in the department of the higher institute of logistics studies (ISEL). In 2021, pursuing further academic growth, Mr Belhomme joined the Research Group on Electrical Engineering and Automatic Control (GREAH) as a Ph.D. student. His research focuses on the innovative hybrid control of a fleet of land and air mobile robots, specifically targeting industrial logistics applications. This work highlights his commitment to integrating advanced robotics into practical industry solutions.



Dr François Guérin got the "Agrégation" in Electrical Engineering (Electronics), an exam to access to civil service in the French public education system.Then, he obtained his Ph.D (2004) and his Habilitation Degree (2015) in Robotics and Automatic Control from Le Havre Normandy University. Currently, he works as Associate Professor within the Higher Institute of Logistics Studies (ISEL) and the Research Group in Electrotechnics and Automatic Control (GREAH) of Le Havre Normandy University. Dr François Guérin teaches Automatic Control, Robotics and Electronics (IoT) and focuses his researches on Advanced Automatic Control applied to Mobile Robotics. From 2021, he is also the President of the Normandy Drone Innovation Center (CIDN).