Available online at www.sciencedirect.com

ScienceDirect

00 (2025) 1–1

Journal Logo

Reconfigurable poly-robots for warehouse transport operations

Mari Chaikovskaia[a], Jean-Philippe Gayon[a,*], Jean-Christophe Fauroux[b,c], Zine Elabidine Chebab[b,c]

[a] *Université Clermont-Auvergne, CNRS, Mines de Saint-Etienne, Clermont Auvergne INP, LIMOS, 63000 Clermont-Ferrand, France*
[b] *MecaBotiX, At Home/ La Cité, 55 Avenue Louis Bréguet, 31400 Toulouse, France*
[c] *Université Clermont Auvergne, Clermont Auvergne INP, CNRS, Institut Pascal, 63000 Clermont-Ferrand, France*

Abstract

We consider a warehouse where loads are transported by a fleet of elementary robots that can be aggregated to form poly-robots. These poly-robots can be dynamically reconfigured to handle various load types. Our work focuses on scheduling this robotic fleet with the goal of minimizing the total time required to complete all transportation operations. We introduce an integer linear programming model for two scenarios, with and without robot reconfiguration. We demonstrate that enabling reconfiguration can greatly reduce execution time in some situations. We also show that the linear programs can be solved for the industrial case study under consideration.

Keywords: Poly-robots; Reconfiguration; Intra-logistics; Warehouse automation; Transportation.

*Corresponding author.
Email address: j-philippe.gayon@uca.fr (Jean-Philippe Gayon)

Available online at www.sciencedirect.com

ScienceDirect

00 (2025) 1–5

Journal Logo

Reconfigurable poly-robots for warehouse transport operations

Mari Chaikovskaia[a], Jean-Philippe Gayon[a,*], Jean-Christophe Fauroux[b,c], Zine Elabidine Chebab[b,c]

[a] *Université Clermont-Auvergne, CNRS, Mines de Saint-Etienne, Clermont Auvergne INP, LIMOS, 63000 Clermont-Ferrand, France*
[b] *MecaBotiX, At Home/ La Cité, 55 Avenue Louis Bréguet, 31400 Toulouse, France*
[c] *Université Clermont Auvergne, Clermont Auvergne INP, CNRS, Institut Pascal, 63000 Clermont-Ferrand, France*

Abstract

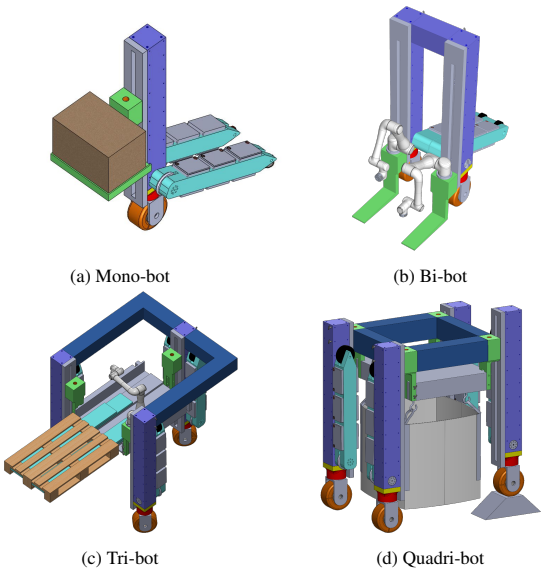
We consider a warehouse where loads are transported by a fleet of elementary robots that can be aggregated to form poly-robots. These poly-robots can be dynamically reconfigured to handle various load types. Our work focuses on scheduling this robotic fleet with the goal of minimizing the total time required to complete all transportation operations. We introduce an integer linear programming model for two scenarios, with and without robot reconfiguration. We demonstrate that enabling reconfiguration can greatly reduce execution time in some situations. We also show that the linear programs can be solved for the industrial case study under consideration.

Keywords: Poly-robots; Reconfiguration; Intra-logistics; Warehouse automation; Transportation.

1. Introduction

A reconfigurable robotic system is a set of modules that can be attached and detached to change shape and adapt to different tasks and environments [1]. For an overview of modular reconfigurable robotics, see [2]. In the context of warehouse logistics and industrial manufacturing, reconfigurable robots offer promising solutions to achieve transportation tasks. Figure 1 shows an example of reconfigurable poly-robots prototyped by MecaBotiX, a company we are collaborating with on this project. The mono-bot can transport a box (or a single product) and pass through a narrow door. The bi-bot has the ability to transport heavier load and to turn easily. The tri-bot can carry a pallet. The quadri-bot is more stable and can carry up to 1000 kg (e.g. a big bag). It can also perform sideways movements, which is unusual for forklifts on the market, or overcome obstacles by using three wheels for support and the fourth one to overcome the obstacle.

It appears that reconfigurable poly-robots can offer several advantages in the future for warehouse transport operations. Firstly, the means of transport can be dynamically adapted to the load size or mass, which prevents to use oversized (or undersized) robots and allows to re-affect the available elementary robots. Secondly, reconfigurable poly-robots may allow to densify storage by using narrower aisles for mono-bots. Thirdly,



(a) Mono-bot (b) Bi-bot

(c) Tri-bot (d) Quadri-bot

Figure 1: M3-Cooper poly-robots, MecaBotiX [3]

*Corresponding author.
Email address: j-philippe.gayon@uca.fr (Jean-Philippe Gayon)

the bots are interchangeable, which improves failure tolerance and reduces maintenance costs.

We review the literature on scheduling mobile manipulator robots within flexible manufacturing systems. The scheduling problem involving a single mobile robot has been studied in works such as [4, 5], while multiple mobile robots have been explored in [6, 7, 8, 9, 10]. Both exact and heuristic approaches have been proposed for (see, for instance, [10] for a comprehensive overview). It is important to note that, in all these studies, only one robot is used at a time for each transportation operation.

Our main contributions are outlined as follows. To our knowledge, this is the first study to deal with the scheduling of a fleet of reconfigurable poly-robots, where elementary robots can be grouped together to form poly-robots and reconfigured over time. We examine two variants of the problem: in the first, poly-robots can be reconfigured before each movement; in the second, reconfiguration is allowed only once, at the start of the time horizon. The goal in both cases is to minimize the makespan, i.e., the total time required to complete all transportation tasks. For each variant, we develop an Integer Linear Programming (ILP) model. These ILP formulations can be efficiently solved in the context of warehouse applications, where the number of load types and robot configurations is limited.

2. Problem description

We consider a fleet of N mobile robots capable of collaborating to transport loads. A p-bot refers to a group of p robots working together to carry loads, with $p = 1, \cdots, P$. For each load type $k = 1, \cdots, K$, there are n_k units to be transported. The planning horizon is divided into T time periods ($t = 1, \cdots, T$), where T is chosen sufficiently large, as discussed later in this section. During any given period, a p-bot can complete a round trip and transport up to c_{pk} loads of type k. It is assumed that each p-bot handles only one type of load per trip. Additionally, all p-bots may be reconfigured at the beginning of each period.

The goal is to minimize the makespan. We consider two scenarios. In the first scenario, poly-robots can be reconfigured at the start of each time period. In the second scenario, poly-robots can be reconfigured only once, at the beginning of the first period. We denote the minimum makespan as T_R when reconfiguration is allowed, and T_W when it is not (except once before the first period). To guarantee the existence of a feasible solution in the non-reconfigurable case, we assume that $N \geq P \cdot K$, ensuring that a poly-robot can be assembled for each load type simultaneously.

To guarantee that we can solve the problem with or without reconfiguration, we must choose a number of periods T such that $T \geq T_W \geq T_R$. To transport all loads of type k by a unique poly-robot, we need no more than n_k periods. Since we can build K poly-robots in each period, thanks to the assumption $N \geq P \cdot K$, it follows that $\max_k n_k$ is an upper bound on T_W. Hence, setting $T = \max_k n_k$, guarantees that we can solve the problems with or without reconfiguration.

3. Integer linear programming formulation

3.1. Reconfiguration at the beginning of each period

We first assume that reconfiguration is possible at the beginning of each period. We use the following decision variables:

- N^t_{pk} : number of p-bots to transport loads of type k in period t;
- α_t : binary variable equal to 1 if some load is transported in period t and to 0 otherwise.

$$T_R = \min \quad \sum_{t=1}^{T} \alpha_t \tag{1}$$

subject to:

$$\sum_{t=1}^{T} \sum_{p=1}^{P} c_{pk} \cdot N^t_{pk} \geq n_k \qquad \forall k \tag{2}$$

$$\sum_{k=1}^{K} \sum_{p=1}^{P} p \cdot N^t_{pk} \leq N \cdot \alpha_t \qquad \forall t \tag{3}$$

$$\alpha_{t+1} \leq \alpha_t \qquad t = 1, \cdots, T-1 \tag{4}$$

$$N^t_{pk} \in \mathbb{N}, \alpha_t \in \{0, 1\} \qquad \forall p, \forall k, \forall t \tag{5}$$

We now comment each line of the above ILP:

- Objective function (1): $\sum_{t=1}^{T} \alpha_t$ represents the makespan;
- Constraint (2): all loads of all types must be transported;
- Constraint (3): the number of bots used in each period must not exceed the fleet size;
- Constraint (4): if period t is inactive, then period $t + 1$ is inactive;
- Constraint (5): the definition domains of variables.

3.2. Reconfiguration only at the beginning of the first period

When reconfiguration is possible only at the beginning of the first period, we need to introduce a new variable N_p, which is the number of p-bots. We also have two additional constraints

in the ILP formulation.

$$T_W = \min \quad \sum_{t=1}^{T} \alpha_t \tag{6}$$

subject to:

$$\sum_{t=1}^{T} \sum_{p=1}^{P} c_{pk} \cdot N_{pk}^t \geq n_k \qquad \forall k \tag{7}$$

$$\sum_{k=1}^{K} \sum_{p=1}^{P} p \cdot N_{pk}^t \leq N \cdot \alpha_t \qquad \forall t \tag{8}$$

$$\alpha_{t+1} \leq \alpha_t \qquad t = 1, \cdots, T-1 \tag{9}$$

$$N_p \geq \sum_{k=1}^{K} N_{pk}^t \qquad \forall t, \forall p \tag{10}$$

$$N \geq \sum_{p=1}^{P} p \cdot N_p \tag{11}$$

$$N_{pk}^t, N_p \in \mathbb{N}, \alpha_t \in \{0,1\} \qquad \forall p, \forall k, \forall t \tag{12}$$

Constraint (10) ensures that the number of robots used in each period does not exceed number of p-bots. Constraint (11) ensures that the fleet size does not exceed the total number of bots N.

4. Reconfigurable vs non-reconfigurable fleet

We can establish the following theorem that compares the makespan with reconfiguration to the makespan without reconfiguration (see proof in Appendix A).

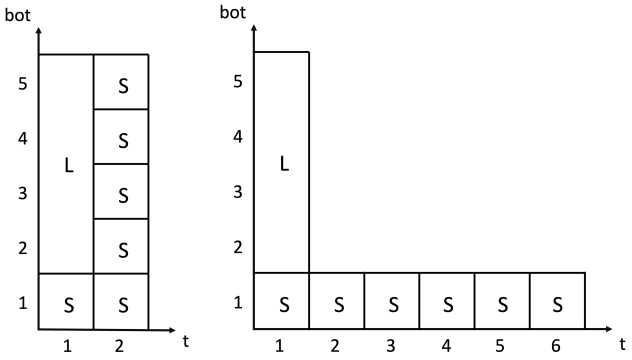
Theorem 4.1. *Let n be the total number of loads to be transported ($n = \sum_k n_k$). When $n \geq 2$, we have*

$$1 \leq \frac{T_W}{T_R} \leq \frac{n}{2}. \tag{13}$$

It can be shown that the upper-bound in (13) is tight for any $n \geq 2$.

We now provide an illustrative example with the following characteristics. Assume that there are two types of loads. The first type of load will be called small (S) and the second type of load will be called large (L). There are $N = 5$ bots, 6 loads S and 1 load L. A 1-bot can carry one load S at a time (and no load L) and a 4-bot can carry one load L at a time (and no load S).

The optimal solutions for both strategies are shown as Gantt charts in Figure 2. When reconfiguration is allowed, a 4-bot transports a load L in period 1 and then splits into four 1-bots in period 2 to transport four loads S. The optimal makespans, with or without reconfiguration, are respectively $T_R = 2$ and $T_W = 6$. Hence the strategy with reconfiguration is 3 times faster than the strategy without reconfiguration for this example.



(a) With reconfiguration (b) Without reconfiguration

Figure 2: Gantt chart for an example with two types of loads

For the same example, if we multiply the number of loads by x, we can show that the strategy with reconfiguration is still 3 times faster than the strategy without reconfiguration.

We conclude that reconfiguration is particularly advantageous in certain contexts, notably when large loads are significantly outnumbered by small ones.

5. Numerical experiments

In this section, we solve the two ILPs introduced in Section 3 with the PuLP library, using CBC (Coin Branch and Cut) solver. The programs are implemented in Python on an Apple M3 Pro 0.7 GHz. We set the maximum computation time to 600 seconds.

5.1. Instances

We build an instance by first choosing N, P, K and next generating the capacities and the demands as follows. For each load type k, we randomly select the capacity c_{1k} in $\{1, \cdots, K\}$. Then we set, $c_{pk} = p \cdot (c_{1k} + \epsilon) + \tau$ with ϵ randomly selected in $\{1, 2\}$ and τ randomly selected in $\{1, 2, 3\}$ for each configuration $p > 1$. We denote by $c_k^{mean} = \frac{1}{P} \sum_p c_{pk}$ the mean capacity. The demand for load type k is then set as $n_k = \lfloor \gamma \cdot J \cdot N \cdot P \cdot c_k^{mean} \rfloor$ where J is randomly selected in $\{1, \cdots, 10\}$. When $\gamma = 0.05$, the order of magnitude of demand is 50 for each load type. When $\gamma = 1$, the order of magnitude of demand is 1 000 for each load type. We also have set $T = \max_k n_k$ as explained at the end of Section 2.

We call nominal instance the instance with parameters ($P = 5$, $K = 5$, $N = 10$, $\gamma = 0.1$). We apply the ILP programs to this nominal instance, then change one parameter at a time among N, P, K, γ, while keeping the others unchanged, as follows:

- $K = 2, 4, 5, 6, 8, 10$
- $P = 2, 4, 5, 6, 8, 10$
- $N = 10, 20, 40, 80, 160$
- $\gamma = 0.05, 0.1, 0.5, 1$

It gives us 18 different instances.

5.2. Results

The results are reported in Table 1. The nominal instance appears in bold in the tables. The optimality gap with the linear relaxation is indicated in brackets for the instances that are not solved to optimality in 600s. The solver finds the optimal solution for all the 18 instances for the problem with reconfiguration and for 16 instances for the problem without reconfiguration. For the last two instances of the problem without reconfiguration, the optimality gap is less than 2 %. For a large number of configurations and load types, other numerical tests show that it is not possible to solve the ILP in a reasonable amount of time. In this case, it would be necessary to develop a heuristic method. However, in practice, most warehouses use only a few types of standardized loads (e.g. boxes and pallets).

N	T_W	CPU time (s)	T_R	CPU time (s)
10	**65**	**19.86**	**64**	**5.47**
20	27	6.83	27	9.01
40	67	252.92	67	9.84
80	44	114.87	44	21.99
160	39	547.32	39	25.49

P	T_W	CPU time (s)	T_R	CPU time (s)
2	2	0.05	2	0.04
4	31	2.69	31	1.14
5	**65**	**19.86**	**64**	**5.47**
6	37	9.19	37	2.31
8	50	9.32	50	1.11
10	174	43.85	174	5.76

K	T_W	CPU time (s)	T_R	CPU time (s)
2	11	0.26	11	0.49
4	33	3.39	33	1.56
5	**65**	**19.86**	**64**	**5.47**
6	63	12.00	62	5.24
8	52	6.31	52	7.92
10	102 (1%)	600	101	9.86

γ	T_W	CPU time (s)	T_R	CPU time (s)
0.05	11	0.78	11	0.28
0.1	**65**	**19.86**	**64**	**5.47**
0.5	223 (1.6%)	600	220	20.33
1	131	28.86	131	4.51

Table 1: Makespan and CPU times with or without reconfiguration

6. Conclusion

We consider a warehouse logistics with standardized loads such as boxes and pallets. Loads are transported by a fleet of elementary robots that can be aggregated to form poly-robot, which can be reconfigured over time. We explain how reconfigurable poly-robots can improve operational efficiency thanks to their adaptability to load type and environment, fault tolerance, stability and maneuverability. In particular, we study the problem of scheduling such a fleet with the objective to minimize the makespan. We propose a mathematical formulation of the problem which can be solved with a linear optimization solver. We demonstrate that incorporating reconfigurability can lead to a substantial reduction in makespan, especially in scenarios where large loads are significantly outnumbered by small loads.

This work has several limitations. Firstly, the ILP approach is not suitable for solving instances involving a large number of load types and configurations. Developing heuristic methods capable of producing approximate solutions with low computation time would be a valuable direction for future work. Secondly, the model assumes that transportation times are independent of load type and configuration. Additionally, robot paths and potential conflicts are not taken into account. Finally, incorporating reconfiguration setup times into the model would provide a more realistic and comprehensive framework.

References

[1] H. Bojinov, A. Casal, T. Hogg, Emergent structures in modular self-reconfigurable robots, in: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings, Vol. 2, 2000, pp. 1734–1741.

[2] J. Seo, J. Paik, M. Yim, Modular reconfigurable robotics, Annual Review of Control, Robotics, and Autonomous Systems 2 (2019) 63–88.

[3] MecaBotiX, `https://www.mecabotix.com/` (2023).

[4] J. Hurink, S. Knust, Tabu search algorithms for job-shop problems with a single transport robot, European journal of operational research 162 (1) (2005) 99–111.

[5] A. Caumond, P. Lacomme, A. Moukrim, N. Tchernev, An MILP for scheduling problems in an FMS with one vehicle, European Journal of Operational Research 199 (3) (2009) 706–722.

[6] L. Deroussi, M. Gourgand, N. Tchernev, A simple metaheuristic approach to the simultaneous scheduling of machines and automated guided vehicles, International Journal of Production Research 46 (8) (2008) 2143–2164.

[7] P. Lacomme, M. Larabi, N. Tchernev, Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles, International Journal of Production Economics 143 (1) (2013) 24–34.

[8] O. T. Baruwa, M. A. Piera, A coloured petri net-based hybrid heuristic search approach to simultaneous scheduling of machines and automated guided vehicles, International Journal of Production Research 54 (16) (2016) 4773–4792.

[9] D. B. M. Fontes, S. M. Homayouni, Joint production and transportation scheduling in flexible manufacturing systems, Journal of Global Optimization 74 (4) (2019) 879–908.

[10] Y.-J. Yao, Q.-H. Liu, X.-Y. Li, L. Gao, A novel MILP model for job

shop scheduling problem with mobile robots, Robotics and Computer-Integrated Manufacturing 81 (2023) 102506.

Appendix A. Proof of Theorem 4.1

Proof. We distinguish two cases.

1. $T_R \geq 2$
   We have $T_W \leq n$ since we transport at least one load per period.
   $$1 \leq \frac{T_W}{T_R} \leq \frac{n}{T_R} \leq \frac{n}{2}.$$
2. $T_R = 1$
   Reconfiguration is not used and $T_W = T_R$. Since $n \geq 2$, it follows that $\frac{T_W}{T_R} = 1 \leq \frac{n}{2}$.

□