A MILP-based heuristic approach for the design of multi-product modular reconfigurable lines

Abdelkrim R. Yelles-Chaouche, Evgeny Gurevsky, Nadjib Brahimi, Alexandre Dolgui

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-10 (2022) 1302–1306

A MILP-based heuristic approach for the design of multi-product modular reconfigurable lines

Abdelkrim R. Yelles-Chaouche * **Evgeny Gurevsky** *
Nadjib Brahimi ** **Alexandre Dolgui** ***

* *LS2N, Université de Nantes, France*
** *Rennes School of Business, France*
*** *IMT Atlantique, LS2N, Nantes, France*

Abstract: This paper deals with the design of a reconfigurable modular manufacturing line. The studied line is composed of a fixed number of modular machines, each of which having a limited number of spots where modules can be placed. Such a line is able to handle different products. Switching from one product configuration to another requires the line to be reconfigured by adding, moving or removing some modules. The objective of this paper is to minimize the total number of modules necessary to produce a given set of products. To do this, a first extended MILP formulation is proposed based on a module generation algorithm. Then, a MILP-based heuristic approach, which uses a module filtering algorithm, is suggested to address the studied problem. Preliminary results and a comparison between the two developed approaches are finally presented and discussed.

Keywords: Reconfigurable manufacturing system, modular line, multi-product, MILP.

1. INTRODUCTION

Modularity is one of the most important characteristics of reconfigurable manufacturing system (RMS), as it enables them to adapt quickly, efficiently and cost-effectively to demand fluctuations and product changes. In fact, modularity aims at dividing the functionalities of a system into interdependent and interchangeable subsets, generally referred to as modules. Each module can then be changed, moved or removed without affecting the rest of the system, thus providing flexibility to adapt in a dynamic environment. A modular architecture of a manufacturing system also helps to better cope with contingencies (such as breakdowns), since the disruption of one module will not have a direct impact on the others. Finally, modular systems are less expensive to develop and modify. This means that instead of developing or modifying an entire complex system, it is sufficient to develop or modify individual modules.

Despite all the advantages of modularity, designing a modular system remains a challenging problem. In the context of production systems, such problem consists of identifying modules needed to manufacture products. Each module can then perform a limited number of tasks. These modules will then be assigned to machines that are compatible from a software and mechanical point of view. In addition, when designing a modular manufacturing system, it is important to find a balance between the optimal number of modules and their ideal size. Indeed, using small modules will result in a less complex system with a large number of modules, while using large modules will decrease their number but increase their complexity.

In this context, we consider a problem of minimizing the number of modules when designing a multi-product reconfigurable production line. It should be noted that few studies have been carried out in this scope where authors consider that a set of modules is already given and aim to select the best ones. In the present paper however, the objective is to find the optimal modules based on precedence constraints and cycle time of products to be manufactured.

More precisely, the studied line consists of a fixed number of modular machines. Each has a limited number of spots where modules can be plugged. Within each machine, modules are activated in a serial way. Moreover, such a line can handle several types of products, each of them characterized by a cycle time and a set of tasks. These latter have to be performed following a partial order usually represented by a directed acyclic graph, called precedence graph. Fig. 1 shows an example of two precedence graphs for two different products. In this figure, the nodes, which represent the tasks, are linked by arcs that express the precedence relations between them. Each task requires a processing time to be completed which is represented at the top of its corresponding node. For each product to be produced, an admissible line configuration, *i.e.*, a line that satisfies precedence graph and cycle time constraints is needed. Thus, switching from one product configuration to another is done by adding, moving and/or removing modules. Fig. 2 illustrates an admissible line configuration for each of the two precedence graphs of Fig. 1, where a total of 9 modules are needed. Among these latter, it is worth noting that modules M1, M2 and M4 can be used for both configurations. As for the other ones, they

10.1016/j.ifacol.2022.09.570

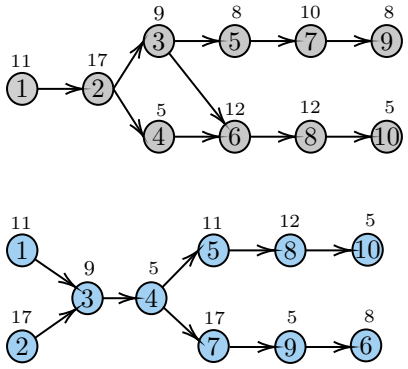

Fig. 1. Two precedence graphs corresponding to two different products

have to be replaced when moving from one configuration to another.

In view of the above described context, an interesting optimization problem, which consists of finding for each product an optimal line configuration that minimizes the total number of used modules, arises. To tackle this problem, we propose a MILP heuristic approach, based on a developed pattern generation algorithm.

2. LITERATURE REVIEW

In this paper, we consider a line balancing and equipment selection problem. It consists of assigning tasks to workstations and the corresponding equipment. Previous studies have tackled this problem. For example, Graves and Redfield (1988) addressed a problem where a family of similar products have to be assembled in a line. The authors proposed a methodology to find a single line configuration that is able to produce all the products. A methodology, that finds the best task assignment and equipment selection, is proposed. The objective is to minimize the total cost, which includes the capital cost related to equipment and variable operating costs. Bukchin and Tzur (2000) propose a branch-and-bound approach to tackle the problem of selecting equipment and assigning tasks to workstations. The objective function minimizes the total equipment costs. Later in Bukchin and Rubinovitz (2003), the authors addressed the same problem by considering the minimization of the number of workstations along with the equipment costs.

The closest problem to the one studied in this paper is from Belmokhtar et al. (2006). Here, the authors considered a modular machining line, which is composed of multi-spindle units. Given a set of available spindle units and their costs, the goal is to design a line by selecting and assigning the chosen spindle-units to machines with respect to precedence, cycle time and technological constraints. An ILP formulation was developed to minimize the investment cost of the selected units. Borisovsky et al. (2013) consider the problem of designing an optimal machining line with respect to task sequence set-up time and several technological constraints. The authors proposed a reduction of this problem to a set partitioning one. As a result, they were able to generate all the possible workstations and find for each of them the optimal sequence of operations that minimizes the total set-up time using dynamic programming approach. Subsequently, three exact approaches were proposed to solve the set partitioning problem that are constraint generation, branch-and-cut and parallel branch-and-cut algorithms. In the same scope, Battaïa et al. (2012) tackle an equipment location problem in a machining line. The authors proposed several pre-processing procedures, based on constraints analysis, in order to reduce the search space. Moreover, they developed an algorithm for generating a lower bound on the number of used equipment.

In view of various studies cited above, one can notice that

- The design and balancing problems of a multi-model line are only addressed on real industrial cases due to the growing interest of such a line structure nowadays. However, there is a lack of generalized studies of such a problem.
- The line balancing problem along with the equipment selection/generation was addressed in the presence of a single product only.

The present paper aim to fill this gap by considering a generic modular multi-product line. The aim is to find an optimal set of modules that can be used to produce different products.

3. MATHEMATICAL FORMULATION

In order to tackle the previously described problem, the mathematical formulation is developed below in order to minimize the total number of used modules necessary to produce a set of known products.

Notations:

- V is the set of all tasks;
- W is the set of available machines;
- P is the set of products;
- M is the set of all generated modules;
- $M^{(k,p)}$ is the set of modules that can be assigned to machine the $k \in W$ of the product $p \in P$;
- α_{im} is equal to 1 if the task $i \in V$ is present in the module $m \in M$;
- $C^{(p)}$ is the cycle time corresponding to product p
- $c_m^{(p)}$ is the load corresponding to module m for product P. It is calculated as the sum of the processing time of the tasks performed by the module.
- $G^{(p)} = (V, A^{(p)})$ is a directed acyclic graph representing the precedence constraints of product p. Here, $A^{(p)}$ is the set of arcs for $G^{(p)}$, where an arc $(i, j) \in A^{(p)}$ means that task j has to be assigned either to the same module as task i, or to succeeding ones.

Decision variables:

- y_m is equal to 1 if the module $m \in M$ is used in at least one configuration, 0 otherwise.
- $\lambda_m^{(k,p)}$ is equal to 1 if module m is assigned to workstation k for product p, 0 otherwise.

$$\min \sum_{m \in M} y_m \tag{1}$$

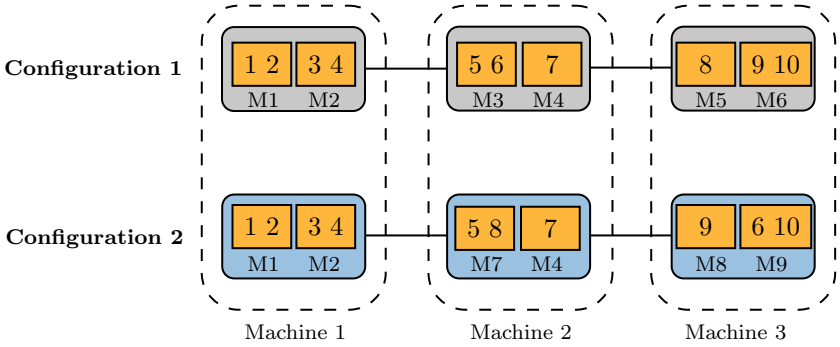

Fig. 2. A modular multi-product line that uses 9 modules

$$\sum_{k\in W}\sum_{m\in M}\alpha_{im}\cdot\lambda_m^{(k,p)}=1,\quad\forall i\in V,\quad\forall p\in P \tag{2}$$

$$\lambda_m^{(k,p)}\le y_m,\quad\forall m\in M,\quad\forall k\in W,\quad\forall p\in P \tag{3}$$

$$\sum_{q=k}^{|W|}\sum_{m\in M}\alpha_{im}\cdot\lambda_m^{(q,p)}\le\sum_{q=k}^{|W|}\sum_{m\in M}\alpha_{jm}\cdot\lambda_m^{(q,p)},\quad\forall(i,j)\in A^{(p)},\ \forall k\in W,\ \forall p\in P \tag{4}$$

$$\sum_{m\in M}c_m^{(p)}\cdot\lambda_m^{(k,p)}\le C^{(p)},\quad\forall k\in W,\quad\forall p\in P \tag{5}$$

$$\lambda_m^{(k,p)}=0,\ \forall m\notin M^{(k,p)},\ \forall k\in W,\ \forall p\in P \tag{6}$$

$$y_m\in\{0,1\},\quad\forall m\in M \tag{7}$$

$$\lambda_m^{(k,p)}\in\{0,1\},\quad\forall m\in M,\quad\forall k\in W,\quad\forall p\in P \tag{8}$$

The objective function (1) minimizes the total number of used modules. Constraints (2) ensure that each task is assigned once in the configuration corresponding to one product. Constraints (3) allow to identify whether a module is used in a configuration or not. Constraints (4) and (5) ensure respectively that precedence and cycle time constraints are satisfied for each product. Finally, constraints (6) are based on module assignment intervals, which allow to set the value of some decision variables. Here, $M^{(k,p)}$ is determined based on the technique developed by Patterson and Albracht (1975) to calculate task assignment intervals.

4. MODULE GENERATION ALGORITHM

The above presented MILP formulation aims to minimize the total number of used modules among a set M of generated ones. Hence, this section presents a pattern generation algorithm used to generate, for each product, all feasible modules.

Given a directed acyclic graph, the main idea of the algorithm is to identify all weakly connected components of a length not exceeding $r_{\max}$, which represents the maximum size of a module. More precisely, the first step of the algorithm consists of determining for each node of the graph its topological level using Kahn's algorithm (Kahn, 1962). Then, the modules are generated by going through

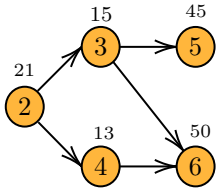

Fig. 3. A example of tasks precedence graph

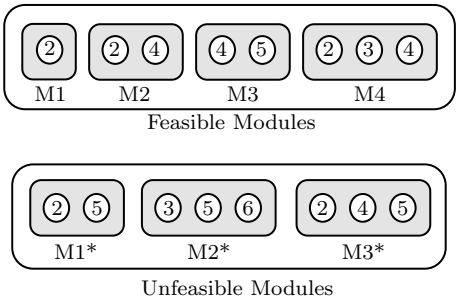

Fig. 4. An example of feasible and unfeasible modules

the nodes of the same topological level as well as through those of following levels. Each time a module is identified, the algorithm ensures that it respects the precedence graph and cycle time of the corresponding product. To better illustrate the process of modules generation, Fig. 3 represents a precedence graph and Fig. 4 shows an example of modules that can be generated and those that cannot be provided because of the aforementioned precedence graph. Here, the cycle time is $C = 100$ and $r_{\max} = 3$.

As a result, modules such as M1, M2, M3 and M4 are generated, whereas modules like M1*, M2* and M3* are not generated since they are infeasible. Indeed, modules M1* and M3* do not respect the precedence graph constraints, and the load of M2* exceeds the cycle time.

5. MILP-BASED HEURISTIC APPROACH

The above presented exact approach is efficient when dealing with small size instances, where the number of generated modules remains relatively small. However, when it comes to medium and large size instances the number of generated modules becomes large, which leads to a combinatorial explosion. To tackle this, we propose a filtering algorithm that significantly reduces the number of modules.

Indeed, when analyzing each optimal solution found for each solved instance, we can notice that the used modules are mainly those which are in common between the configurations. This means that one module can be used to produce the different products. The filtering algorithm consists then in removing modules that are not in common between all the products. More precisely, let $M^{(p)}$ be the set of generated modules for the product $p \in P$, then the set of all generated modules becomes $M = \bigcap_{p \in P} M^{(p)}$, meaning that M only contains common modules.

To reinforce this assumption, the following constraints are added to the above described MILP formulation.

$$\sum_{k \in W} \lambda_m^{(k,p)} \geq y_m, \quad \forall m \in M, \quad \forall p \in P, \tag{9}$$

where constraints (9) ensure that if a module is assigned to a configuration, then it should be also assigned to all the remaining ones.

Fig. 5 shows an example of two feasible solutions. Using the filtering algorithm, the first solution will never be generated because it uses 4 different modules. Rather, the algorithm will propose solutions of the same type as the second solution where all modules are in common and the total number of used modules is least.

This approach is very efficient as it can considerably reduce the search space as well as the solving time. As it will be shown in the next section, the filtering algorithm is able to find optimal solutions of all the instances tested so far. However, this remains a heuristic.

6. COMPUTATIONAL RESULTS

To test the above described MILP formulation and heuristic approach, we used small size instances of 20 tasks ($|V| = 20$) provided by Otto et al. (2013). Two and three products are considered with $r_{\max} = 2$, $r_{\max} = 3$ and $r_{\max} = 4$. Moreover, instances were grouped into three categories based on the density of their precedence graph, generally referred to as order strength (OS). Thus, the first (resp. second and third) category includes the instances having an $OS \approx 0.2$ (resp. $OS \approx 0.6$ and $OS \approx 0.9$). In addition, for each instance the number of machines is fixed to $\max_{p \in P} \left\{ \left\lceil 1.2 \cdot \sum_{i \in V} t_i^{(p)} / C^{(p)} \right\rceil \right\}$, the cycle time $C^{(p)}$ is set to $\left\lceil 1.5 \cdot \max_{i \in V} t_i^{(p)} \right\rceil$ for each product $p \in P$, and the maximum CPU solving time is set to 600 seconds.

The experimental results were obtained using ILOG CPLEX solver 12.10.0 installed on an 1.90GHz Intel(R) Core(TM) i7-8650U computer with 32 GB RAM. The results of the MILP formulation are expressed in Table 1. In this latter, the first column represents the number of tasks, and the second one shows the maximum number of tasks per module. The third and fourth columns express the number of products and the OS category, respectively. The total number of instances of each category is shown on the fifth column. The sixth and seventh columns indicates the number of instances solved to optimality and their corresponding average CPU time, which is composed of the time to generate the modules and the solving time by the solver. Finally, the average GAP for instances that were not solved within 600 seconds is displayed in the last column.

The preliminary results from Table 1 show that all the instances were solved to optimality in less than 7 seconds on average. However, it is interesting to notice that the CPU time has significantly increased when solving instances of $r_{\max} = 4$ and $|P| = 3$. This is due to the fact that the number of generated modules increases with the instance size. This can be confirmed when dealing with medium size instances with $|V| = 50$, where the solver is not able to provide any feasible solution because of the large number of generated modules.

Table 2 shows a numerical comparison between the results described in the previous table and those obtained using the heuristic approach on the same set of instances. In this table, the six first columns are the same as Table 1. As for the last column, it shows the average GAP from the optimal solution, noted as $\mathtt{GAP}_{\mathtt{OPT}}$. The latter is calculated as $\mathtt{GAP}_{\mathtt{OPT}} = \frac{(\mathtt{S_H} - \mathtt{S_{OPT}})}{\mathtt{S_{OPT}}} \cdot 100\%$, where $\mathtt{S_H}$ is the solution found by the heuristic, and $\mathtt{S_{OPT}}$ is the optimal solution found with the MILP formulation.

From Table 2, it can be seen that the heuristic is efficient, since it is able to find all optimal solutions as the MILP formulation. The average CPU time of the heuristic is also promising as it remains below 1 second on average even for instances corresponding to $r_{\max} = 4$ and $|P| = 3$.

7. CONCLUSION AND PERSPECTIVES

This paper addressed the design problem of a reconfigurable modular manufacturing line. Such a line is composed of a fixed number of modular machines and is able to produce a given set of products. Hence, given the precedence graph and the cycle time of each product, the objective is to minimize the total number of used modules necessary to produce all the products. For this purpose, a module generation algorithm was developed in order to generated for each product all feasible modules. Then, a MILP formulation that minimizes the number of modules was presented. Finally, a new MILP-based heuristic was proposed to efficiently solve large size instances. This approach is based on a module filtering algorithm, which aim is to reduce the number of feasible modules. Preliminary numerical experiments showed that the MILP formulation is efficient when dealing with small size instances. However, this approach reaches its limit when dealing with medium size instances when the number of generated modules significantly increases. A numerical comparison between the sole MILP and the MILP-based heuristic showed promising results, since the latter one was able to find optimal solution for all the instances.

The next steps of this work is to asses the performance of the heuristic approach on medium and large size instances. Moreover, it can be interesting to extend this problem on real industrial cases that are available in the literature.

REFERENCES

Battaïa, O., Gurevsky, E., Makssoud, F., and Dolgui, A. (2012). Equipment location in machining transfer lines with multi-spindle heads. *Journal of Mathematical Mod-*

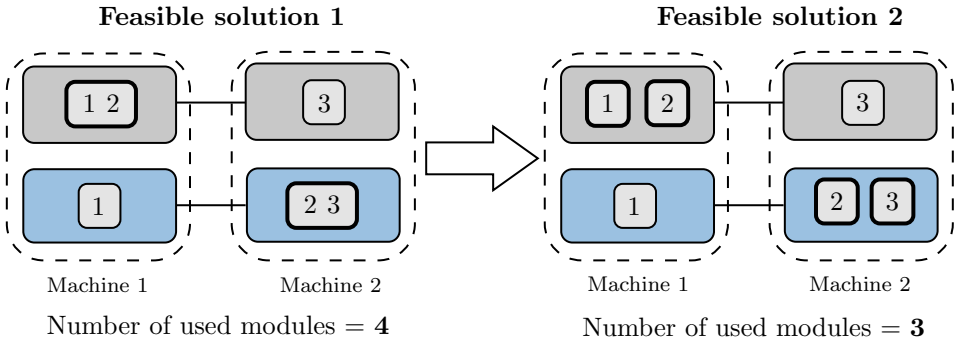

Fig. 5. Filtering algorithm

Table 1. Computational results for the MILP model.

$\|V\|$	$r_{\max}$	$\|P\|$	OS	#Instances	#OPT	Avg. CPU, (s.)	Avg. GAP, (%)
20	2	2	1	3	3	0.64	⊕
			2	199	199	0.50	⊕
			3	68	68	0.18	⊕
		3	1	2	2	1.11	⊕
			2	193	193	1.37	⊕
			3	65	65	0.36	⊕
	3	2	1	3	3	0.72	⊕
			2	199	199	1.28	⊕
			3	68	68	0.25	⊕
		3	1	2	2	1.18	⊕
			2	193	193	2.43	⊕
			3	65	65	0.50	⊕
	4	2	1	3	3	4.23	⊕
			2	199	199	6.48	⊕
			3	68	68	0.55	⊕
		3	1	2	2	13.91	⊕
			2	193	193	45.66	⊕
			3	65	65	2.06	⊕

(⊕) All optimal solutions were found within the time limit of 600 seconds.

Table 2. Computational results for the heuristic approach.

$\|V\|$	$r_{\max}$	$\|P\|$	OS	#Instances	Avg. CPU, (s.)	Avg. GAP_{OPT}, (%)
20	2	2	1	3	0.46	0.00
			2	199	0.35	0.00
			3	68	0.13	0.00
		3	1	2	1.03	0.00
			2	193	0.90	0.00
			3	65	0.23	0.00
	3	2	1	3	0.58	0.00
			2	199	0.86	0.00
			3	68	0.20	0.00
		3	1	2	0.79	0.00
			2	193	1.24	0.00
			3	65	0.23	0.00
	4	2	1	3	1.23	0.00
			2	199	1.10	0.00
			3	68	0.18	0.00
		3	1	2	1.76	0.00
			2	193	2.17	0.00
			3	65	0.42	0.00

elling and Algorithms in Operations Research, 12(2), 117–133.

Belmokhtar, S., Dolgui, A., Guschinsky, N., and Levin, G. (2006). Integer programming models for logical layout design of modular machining lines. *Computers & Industrial Engineering*, 51(3), 502–518.

Borisovsky, P., Delorme, X., and Dolgui, A. (2013). Balancing reconfigurable machining lines via a set partitioning model. *International Journal of Production Research*, 52(13), 4026–4036.

Bukchin, J. and Rubinovitz, J. (2003). A weighted approach for assembly line design with station paralleling and equipment selection. *IIE Transactions*, 35(1), 73–85.

Bukchin, J. and Tzur, M. (2000). Design of flexible assembly line to minimize equipment cost. *IIE Transactions*, 32(7), 585–598.

Graves, S.C. and Redfield, C.H. (1988). Equipment selection and task assignment for multiproduct assembly system design. *International Journal of Flexible Manufacturing Systems*, 1(1), 31–50.

Kahn, A.B. (1962). Topological sorting of large networks. *Communications of the ACM*, 5(11), 558–562.

Otto, A., Otto, C., and Scholl, A. (2013). Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. *European Journal of Operational Research*, 228(1), 33–45.

Patterson, J.H. and Albracht, J.J. (1975). Assembly-line balancing: Zero-one programming with Fibonacci search. *Operations Research*, 23(1), 166–172.