Production resetting optimization (Problem definition)

Context:

In many factories, the production lines can only produce one type of product at time. In order to change production from one product type to another, it is often necessary to change the tools on each machine of the line: in scheduling problems, this is known as setup times. Thus, the products of the same types can be aggregated in batches to avoid doing too many setup. On the other hand, it can also be necessary to split batches if we want to deliver the products on time to customers and to keep low inventory level of finished products.

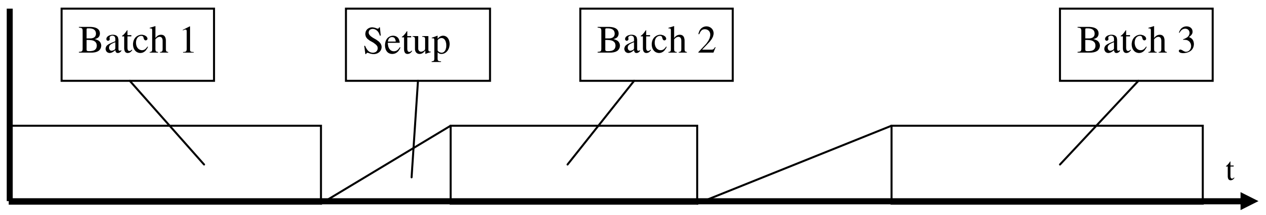


Figure 1. Sequence of batches with setup times between them

Usually the factory should always do a tradeoff between doing batches as large as possible and maximizing the production and doing small batches and deliver products quicker to the customers. It is easy to see that the shorter the setup times, the easier it is to split batches. Therefore, it is very import to reduce setup time as much as possible to improve production flexibility. This is the goal of our work: we propose methods to schedule efficiently the tasks done by operators during a setup.

This study has been done in collaboration with the SKF factories that are producing ball bearings. Therefore, all assumptions are done with these factories in mind.

Problem definition

A production line is compound of n machines. Each piece must go through all machines of the line.

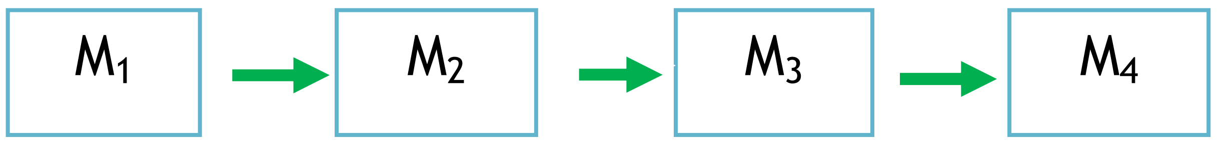


Figure 2. A production line compound of 4 machines

During a production resetting, all the machines should be stopped; then, the tools used by the machines are changed and set up before restarting the machines. The production can only be restarted once all the machines are set up.

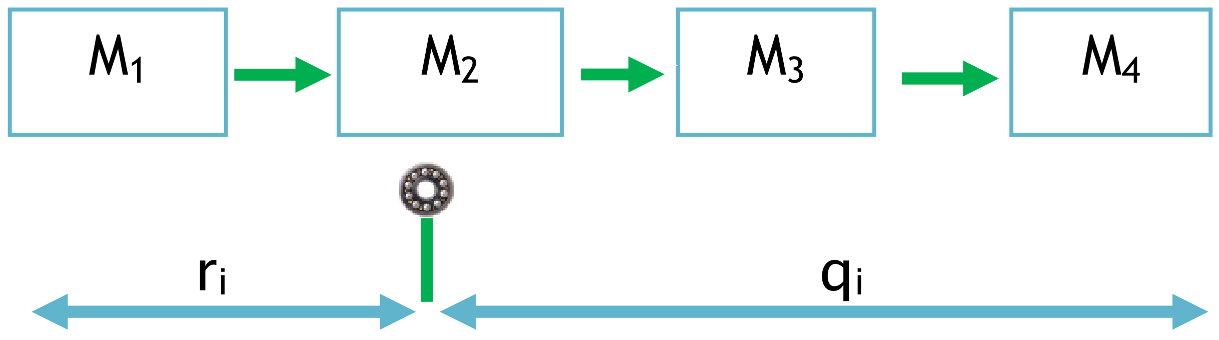
A production resetting can be compared to a pit stop in a formula 1 race: this is time that has to be wasted and it is compound of several operations like changing tires, refueling tanks… The car can only restart when all these operations are finished. Even if a pit stop only lasts a few seconds during a 2 hours long race, it is a very decisive action. Exactly like pit stops, a production resetting is short when compared to production of batches, but it can really make the difference. Low production loss during a production resetting is actually considered as a decisive advantage over other competitors and is the key to just in time production policy as described in the Kanban strategy used by Toyota factories [Shingo 1985].

Each setup task must be done by an operator. In our case a setup task is a complex operation that requires experience: the time needed by an operator to setup a machine highly depends on his experience, skills, preferences… This is why we are considering different processing times for each machine/operator couple.

Moreover, the tasks can only be performed when a machine is empty: all pieces of the current lot must have been processed by the machine. It means that tasks can only be performed after a release date r_i. Similarly, the goal of the problem is to minimize the completion date C_{max} of the first piece: this is when the first piece of the new batch reaches the last machine. To model this we define a delay q_i after each task before it can be considered totally finished.

In scheduling, such problem is known as an unrelated parallel machine problem with release dates and tails: the resources of the problem are the operators, and the tasks are the setup operations.

Example:



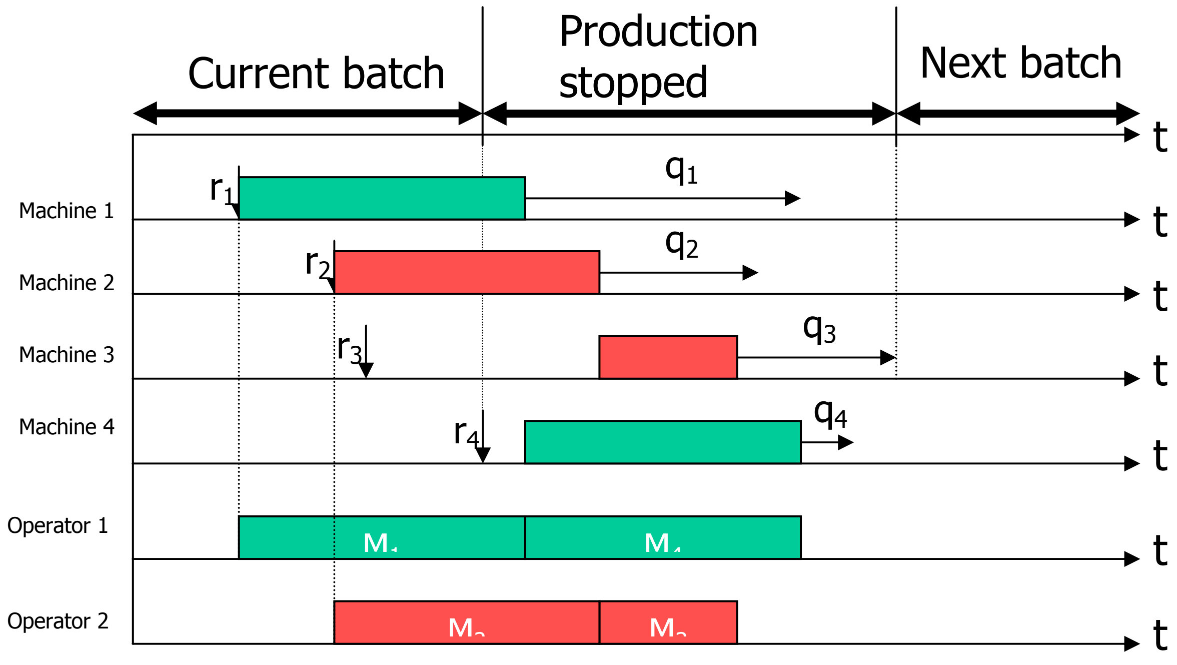


Figure 3. An example of setup tasks scheduling during a production resetting

In this example, the production line is compound of 4 machines and 2 operators are available to perform the setup tasks. The first operator setup the first machine M_1 after it is empty (that is after the date r_1) and then the machine M_4. The second operator setup the machine M_2 after r_2 and then the machine M_3. After the last machine M_4 is empty, that is after the date r_4, the production is stopped. When the first ball bearing of the new batch reaches the end of the production line (that is when the last delay q_i is elapsed) the production of the new batch can begin. This is this date that we try to optimize.

Solution approach

The method we propose to solve this problem hybridizes two classical methods: a Branch-and-Bound and a genetic algorithm. In the following sections, we describe these two methods and how the methods are collaborating together.

Branch-and-Bound:

A Branch-and-Bound is an exact method that enumerates implicitly all possible solutions of a problem to find an optimal one. We name the set of all possible solutions the search space of the problem. In order to use a Branch-and-Bound method, we represent the search space with a tree. Each node of the tree represents a decision that should be taken inside the problem.

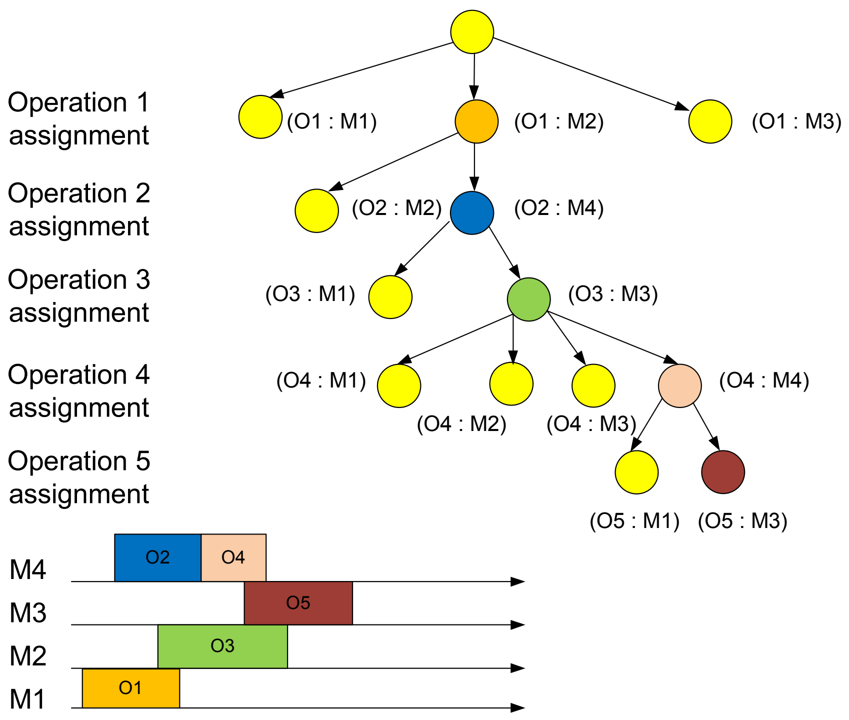


Figure 4. Branching scheme: an operation is assigned to an operator on each node

On this example applied to our problem, the decisions that should be taken are the assignments of tasks to operators. Within each node of the search tree, a task is assigned to an operator.

Going from one level of the search tree to another one is equivalent to splitting the search space: each node represents a little part of the search space.

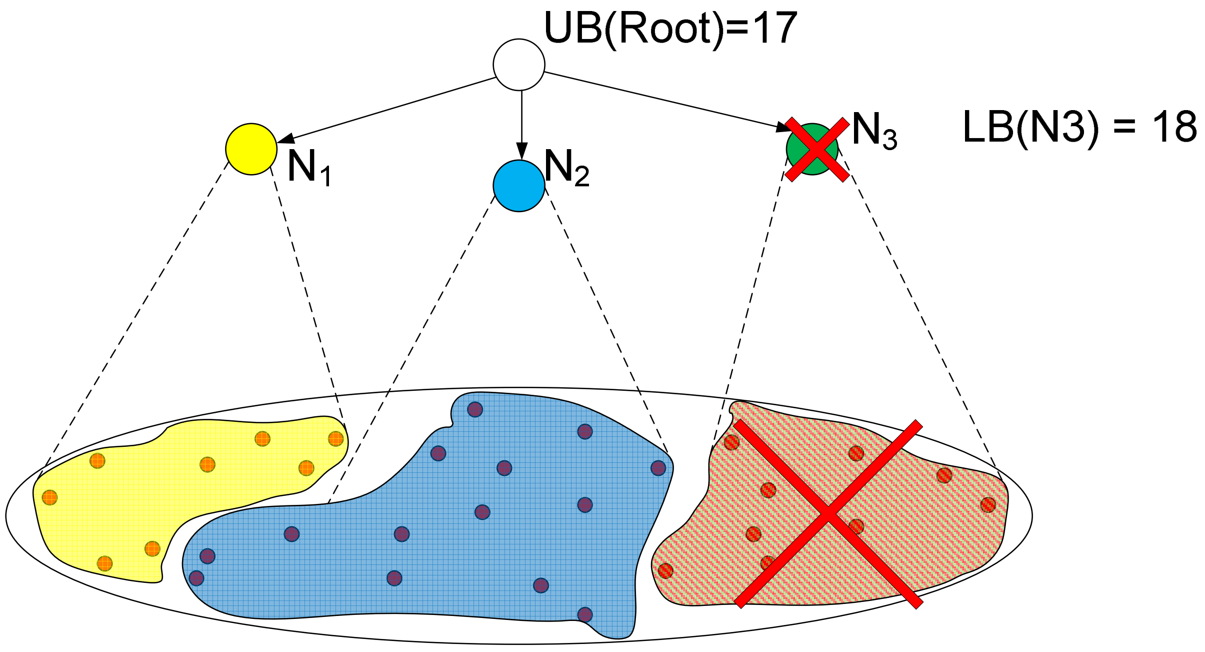


Figure 5. Search space is divided when going from one level of the search tree to another one

On this example, the root node represents the whole search space; this is the first level of the search tree. When we develop the tree and create the second level of the search tree, we take the first decision of the problem. On this example, the first task can be assigned to three operators who have the necessary skills; it means that there are three nodes on the second level corresponding to each of these operators. Those three nodes also represent distinct areas of the search space.

The main idea used in Branch-and-Bound methods is to use bounds to not explore the whole search tree by cutting nodes as soon as we detect that a node can not lead to a good solution. To do this we define an upper bound that is the best known solution that has been found up until now. Then, each time a node is explored, we evaluate it by calculating a lower bound of all the solutions contained in the part of search space represented by the node. If the lower bound is worse than the upper bound, we can cut the node. Otherwise, we can expand the node and look more precisely into this part of the search tree.

On our problem, a lower bound can be obtained by allowing preemption of the tasks. It means that a task can be stopped and restarted later on the same operator or another operator. Such problem is less constrained as we do not force tasks to be executed during a single continuous period on only one operator and it is usually easier to solve. It has been shown by Lawler and Labetoulle [Lawler 1978] that when preemptions are allowed, this problem is easy to solve using a well known method: linear programming. To use this method, the problem should be mathematically modelled using only linear constraints and objectives; this model is then solved using one of the linear program solvers available on the market, as long as the model only uses continuous variables (and that is the case for (and that is the case for the preemptive relaxation), these solvers find the optimal solution with a short time. More details on linear programming can be found in [Gartner 2006].

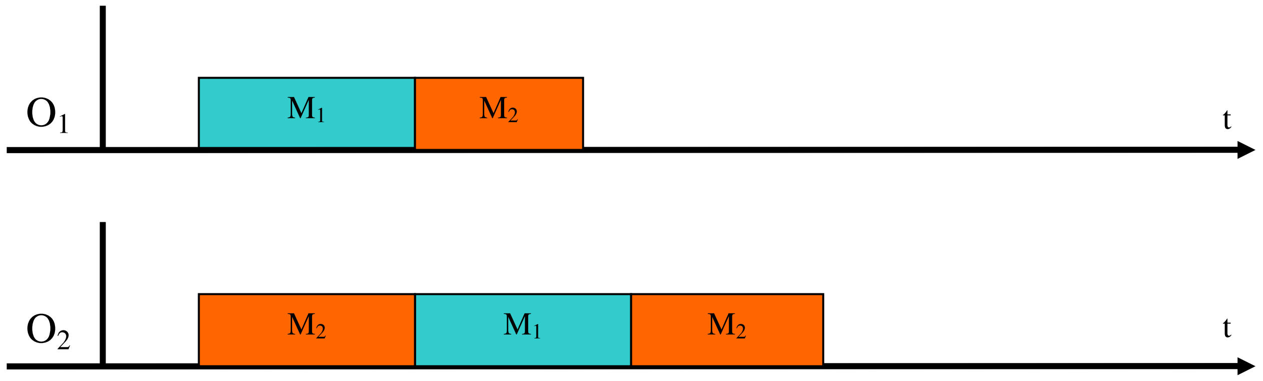


Figure 6. Example of solution with preemption

Genetic algorithm:

A genetic algorithm is a method that uses evolution rules similar to natural phenomenoms that can be observed in nature. We define a population of solutions generated randomly. Then, during the execution of the algorithm, this population will evolve and generation after generation only the best solutions will survive.

To imitate natural evolution mechanisms, we encode solutions into what we call a chromosome. The chromosomes are then crossed over and mutated randomly and the best solutions have more chances to survive (cf. figure 7).

This kind of algorithm has been proven to converge slowly toward the optimal solution. However, it usually finds some good solutions very quickly. Moreover, it is interesting to remark that the time required to find the optimal solution depends on the size of the search space it is exploring.

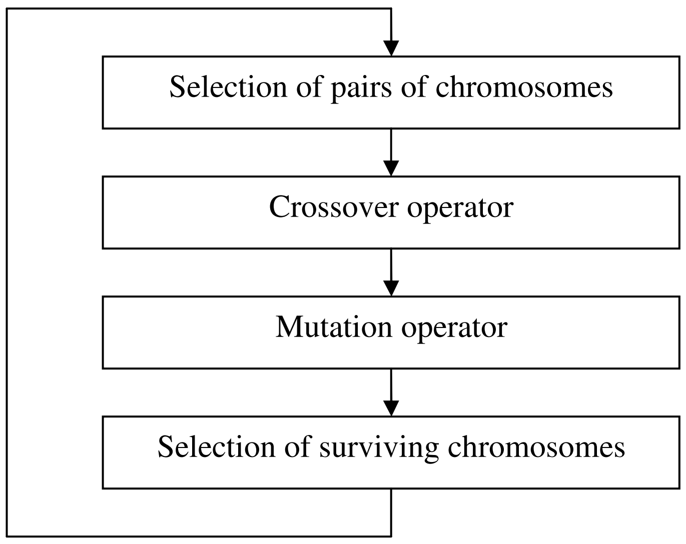
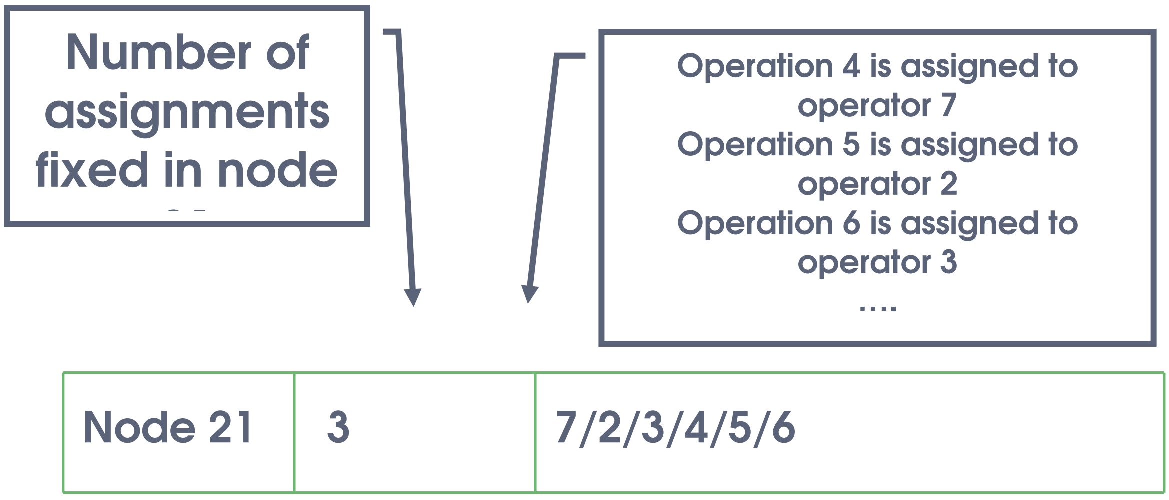

Selection of pairs of chromosomes
Crossover operator
Mutation operator
Selection of surviving chromosomes

Figure 7. Main steps of a genetic algorithms

Cooperation between both methods:

In this method, both the Branch-and-Bound and the genetic algorithm are collaborating together. The idea is that a genetic algorithm is usually able to find good solutions in a short time. This good solution will help the Branch-and-Bound to prune more nodes in the search tree. On the other hand, when the Branch-and-Bound prune nodes, it decreases the size of the search space that should be explored by the genetic algorithm, and with a limited search space, the genetic algorithm can quickly find better solution that will help the Branch-and-Bound and so on…

Both methods are running in parallel and are therefore more efficient on new multi core processors used widely in most modern computers. The main difficulty to implement such method is that the genetic algorithm should be forced to only search in the space that has not been explored yet by the Branch-and-Bound. This can be achieved using the encoding method presented on the figure 8.


Number of assignments fixed in node
Operation 4 is assigned to operator 7
Operation 5 is assigned to operator 2
Operation 6 is assigned to operator 3
....
Node 21
3
7/2/3/4/5/6

The figure 8 represents the way we encode a chromosome in the genetic algorithm. The first field represents a node not explored by the Branch-and-Bound. The second field represents the number of assignments contained within the node. The last field contains the assignments that complete the node in the solution represented by the chromosome. This way, we only have to check that the node is within the list of the non explored nodes of the Branch-and-Bound method.

The crossover operator creates a chromosome from two parent chromosomes. It copies a random number of assignments from the first parent chromosome and the remaining assignments from the second parent chromosome. The mutation operator simply changes a random assignment.

Conclusion:

This method has been tested on a single core machine and we have found that it provides better performances than using the Branch-and-Bound only despites the fact that the processor time is shared by both methods. Moreover, it is quite interesting in practice because it provides good solutions quickly and can give some idea on how far the best solutions are from the optimal. The decision maker can then decide to wait for the optimal solution or to stop computation because the solution is good enough. More details on this method and experimental results can be found in [Pessan 2008a].

This work has been done during a PHD thesis [Pessan 2008b] at the University François Rabelais Tours. The method presented here has been implemented in the SKF France factory of Saint-Cyr-sur-Loire and is used daily to prepare production resetting. It has been shown to provide efficient solutions in practice.

Further work is currently being made on this method. We are working on adding additional conditions to the lower bound used by the Branch-and-Bound: the idea is to improve the lower bound quality and to help the Branch-and-Bound to prune more nodes.

References:

[Gartner 2006] B. Gärtner, J. Matoušek, Understanding and Using Linear Programming, Berlin: Springer. ISBN 3-540-30697-8, 2006

[Lawler 1978] E.L. Lawler, J. Labetoulle. *On preemptive scheduling of unrelated parallel processors by linear programming.* Journal of the ACM, vol. 4, no. 25, pages 612-619, 1978

[Pessan 2008a] C. Pessan, J.L. Bouquard, E. Néron. Artificial evolution, isbn:978-3-540-79304-5, volume 4926/2008 of *Lecture Notes in Computer Science*, chapter *Genetic Branch-and-Bound or exact genetic algorithm ?*, pages 136-147. Springer Berlin / Heidelberg, 2008

[Pessan 2008b] C. Pessan, *Optimisation de changements de séries par ordonnancement des tâches de réglage*, PhD thesis, Université François Rabelais Tours, France, November 2008

[Pessan 2008c] C. Pessan, J.L. Bouquard, E. Néron, *An unrelated parallel machines model for an industrial production resetting problem*, European Journal of Industrial Engineering, vol. 2, no. 2, pages 153-171, 2008

[Shingo 1985] S. Shingo. A revolution in manufacturing: the smed system. Cambridge, MA: Productivity Press, 1985