Dynamic Rebalancing of an Assembly Line with a Reachability Analysis of Communicating Automata

MANCEAUX Antoine[1, 2, 3], BRIL EL-HAOUZI Hind[1, 2], THOMAS André[1, 2], PÉTIN Jean-François[1, 2]

[1] Université de Lorraine, CRAN, UMR 7039, Campus Sciences, B.P. 70239, Vandœuvre-lès-Nancy Cedex 54506, France
{antoine.manceaux, hind.el-haouzi, andre.thomas, jean-francois.petin}@univ-lorraine.fr
[2] CNRS, CRAN, UMR 7039, France
[3] TRANE SAS, rue des Amériques, 88190 Golbey, France
Antoine_Manceaux@trane.com

Abstract. This article proposes a method for dynamically rebalance an assembly line when disturbances occur, by reassigning the tasks to the line's workstations. The method is based on reachability analysis of an automata network that represents the tasks and workstations to be performed. The execution trace leading to the desired state provides one feasible solution to rebalance the assembly line. The method is illustrated by an industrial case study.

Keywords: Assembly Line Balancing · Dynamic · Reconfiguration · Automata · Discrete Event Systems · Reachability analysis

1 Context

Assembly lines are flow-oriented production systems. They are still typical in industrial production systems of high quantity standardized products. In this kind of systems, the problem of properly assigning operations to workstations is called assembly line balancing problem (ALBP). The ALBP is older than 1960 and has been tackled by Operational Research over several decades as can be seen in surveys [1, 2]. Furthermore, several classifications were proposed for this kind of problem, [3, 4] contributing to fill the gap between real problem and academic ones [5].

Scholl, in 1999, [6] gives three levels in line balancing problems that correspond to long and medium-term decisions in case of yet-to-be-built assembly line for a 2 - 5 years horizon, line re-engineering for 6 months - 2 years horizon (for example in [7, 8]), and rebalancing engineering due to a market dimension change for a 1 month - 1 year horizon. This classification does not deal with short planning horizon balancing. Indeed, for this horizon (less than 1 month) the decision is mainly made on scheduling or sequencing decisions (master scheduling or daily sequencing) rather than a line rebalancing. For example, the 2005 ROADEF's Challenge aim was to find a solving

algorithm car sequencing which better fits the existing line balancing for a daily production objective [9].

Our objective here is to introduce dynamic rebalancing for short time horizon when disturbances occur, such as shortage, shutdown or when the theoretic production durations differ from the realized ones. We assume that the manual assembly line is initially balanced (computed by a predictive balancing process) and the sequencing is fixed. To face the disturbances, a modification of the tasks' assignment will be proposed in such a way that the line is kept balanced. The aim is to quickly react to disturbing events with an on-line algorithm even if the new obtained balancing is not optimal. To initiate this kind of on-line dynamic rebalancing, real-time information about the work in progress must be available, leading to put this study in the intelligent manufacturing systems context (IMS) where resources and products can share and update their own data.

This paper explores the use of communicating automata to deal with dynamic rebalancing of a manual assembly line. In section 2 a formal description of the problem is presented. Following, the reachability analysis to resolve an ALBP is presented and explained in section 3. An industrial application from Trane Company and its results are discussed in section 4. Finally, section 5 concludes and displays some future works.

2 Problem Formalization:

A well-balanced assembly line is one where all the workstation loads are smoothed with a working time very close to the takt time. It is defined by the available time divided by the number of products to do. This takt time leads to define a moving frequency and synchronization events where products move from a workstation to the next one. The following section provides some notations for the balancing problem that is addressed by the paper.

2.1 Data

- $T = \{t_i, i \leq Tmax \in \mathbb{N}\}$ is the set of tasks (t_i is the task identifier and Tmax the number of tasks).
- $Dt = \{dt_i \in \mathbb{N}, i \leq Tmax \in \mathbb{N}\}$ is the set of task durations.
- $W = \{w_i, i \leq Wmax \in \mathbb{N}\}$ the set of workstations (w_i is the workstation identifier and Wmax the number of workstations).
- $Dw = \{Dw_i, \in \mathbb{N}, i \leq Wmax \in \mathbb{N}\}$ is the set of workstation durations, that is defined by the sum of task durations that are assigned to this workstation.
- $E = \{e_i, i \leq Emax \in \mathbb{N}\}$ is the set of synchronization events, *i.e.* the instants when the products change of workstations (e_i is the event identifier and Emax the number of events in the studied period).

2.2 Constraints and objective

Assignment is given by a surjective function A: T → W that defines for each task $t_i \in T$, the workstation $w_k \in W$ t_i where is assigned: $A(t_i) = w_k$ (one task must be assigned on one and only one workstation, a workstation can host several tasks). Two kinds of constraints must be fulfilled for a proper line rebalancing:

- **Takt Time (C1):** The assignment of the task t_i to the workstation w_k is possible only if the remaining available capacity of w_k (takt time minus the sum of assigned task durations dw_k) is upper or equal to the task duration dt_i.
- **Precedence (C2):** this constraint is given by the precedence matrix P where P_{ij} equals "1" if t_i must precede t_j, 0 otherwise, for a couple of tasks $(t_i, t_j) \in T^2$.

The reconfiguration (rebalancing of the assembly line) consists in finding one feasible solution (a new task assignment "A_w") that respects the previous constraints and with a short computing time compliant with the workshop time scale.

2.3 Related works

Dynamic rebalancing problem can be addressed by traditional constraint solving methods using scheduling and operational research theories. Due to its complexity, most of the addressed solving approaches are based on metaheuristics ([7, 8]). Even if these approaches are efficient in engineering steps, their computing time is often not compliant with production time scale when applied for purely reactive solutions.

Faced to these classical approaches, methods based on Discrete Event Systems (DES) theory are emerging to model and solve scheduling problems. More particularly, the efficiency of Timed Automata (TA) and reachability analysis techniques have been demonstrated by [10] and [11]. The basic underlying idea is to use reachability analysis and model-checking tools [12] in order to find a possible path for reaching an expected state (*i.e.* the state where all the tasks has been reassigned in such a way the line is kept balanced). The trace from initial state to the expected state provides one admissible balancing solution. Main benefits of DES approaches rely on the modular and parametric way of modeling, and finally, the ability to find feasible solutions with a computing time that is compliant with on-line constraints.

3 Using Reachability Analysis for rebalancing

Our approach is based on two models using a set of communicating automata:

- The task model TM defines the tasks that have to be assigned,
- The workstation model WM defines the ability of a workstation to accept an assignment, taking into account the constraints C1 and C2.
- The synchronization between task and machine models is supported by a competing request/answer mechanism [13].

3.1 Used Formalism

Communicating Automata are a subclass of the Timed Automata formalism defined by Alur and Dill in 1994 [14] that share variables and are synchronized by transition labels. A communicating automaton A is an N-tuple A = (D, X, L, T, Q_m, q_0, v_0), where:

- Q is a finite set of locations;
- X is a finite set of integer variables;
- L is a set of synchronization labels, decomposed into three separated sets: reception labels (noted *label?*), emission labels (noted *label!*) and local labels;
- T is a set of transitions (q, l, g, m, q') ∈ Q × L × G × M × Q where G is the set of guards (conditions on the variables of X) and M is the set of updates of the valuations of variables; l, g and m are optional but a transition must contain at least a label or a guard;
- $Q_m \subseteq Q$ is the set of marked locations;
- $q_0 \in Q$ is the initial location;
- v_0: X ← ℕ is the initial valuation of the variables.

A network NA = A_1 || A_2 || … || A_n of n (n ∈ $\mathbb{N}^*$) is defined as the synchronous product of all the A_i automata. A state of the network is defined by a couple (q; v) where q ∈ Q and v ∈ X. Two kinds of evolution of the automata network NA(q, v) $\xrightarrow{t}$ (q', v0) may occur:

- only one transition is fired in one automaton, if this transition contains only local label or if its guard is satisfied;
- two transitions t_k^γ, t_m^β of a pair of automata (A_γ , A_β) with t_k^γ containing the emission label $l_k^\gamma \in L^\gamma$ (noted l_k^γ!) and t_k^β containing the emission label $l_m^\beta \in L^\beta$ (noted l_m^β?) such that $l_k^\gamma = l_m^\beta$ are fired simultaneously, providing that the guards of these transitions are satisfied.

Note that simultaneous firing of transitions is possible only when two transitions of two different automata are considered; no broadcast mechanism that implies more than two automata is possible. Notation conventions are as follows: initial locations are indicated by a source arc, location names are in bold, label names are in italics and followed by the symbol "!" (resp. "?") for emission (resp. reception) labels, variables updates are underlined, and guards are denoted by brackets.

Task Generic Model.
The generic task model TM (Figure 1.a) defines a task t_i which has to be assigned and is composed by three locations.

In the initial location, the task t_i is waiting for an assignment on a workstation w_k. The transition that can be fired corresponds to the emission of an assignment request on the workstation w_k.

Once this request has been emitted, the model is waiting in the "*waiting for a workstation answer*" location for an answer from a workstation model that can be:

- a refusal: in this case, the task comes back in its initial location, ready for another possible question;
- an acceptance: in this case, the template attains the last location;

The last location "*task assigned*" represents an assigned task which could not make another request (uniqueness of the assignment).

Workstation Generic Model.

The generic workstation model WM (Figure 1.b) is composed by two locations. It defines a workstation w_k which accepts or refuses task assignments according to defined constraints (C1 and C2). In the initial location, the workstation w_k is waiting for an assignment request. Once a request is received from a task model TM, the location called "*Computing answer*" is reached. From this location "*Computing answer"*, there is two transitions with exclusive guards containing the two constraints depending from the current workstation capacity:

- if C1 and C2 are false, the workstation rejects the assignment by sending a refusal to the task t and returns to its initial location.
- if C1 and C2 are true, the workstation accepts the assignment by sending an acceptation message.

If the request is accepted, the assignment parameters are recorded as the list of already assigned with $A(t_i) = w_k$.

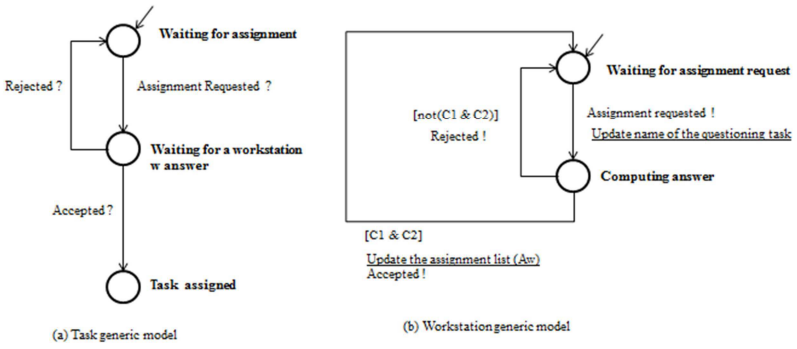


Fig. 1. Generic models of tasks (a) and workstations (b)

The Assembly Line Generic Model and the Initialization of the Model.

The complete model is a network of communicating automata that is composed of (Tmax – m) instances of the task model (where m represents the number of tasks that are already finished when the rebalancing is done), and Wmax instances of the workstation model. The initial capacity of workstations is set according to the already finished tasks.

The correct synchronization is ensured by sending and receiving message *("Assignment Requested", "Rejected", "Accepted"*). To avoid inconstancies (the sender task

must be the same that the one who receive the workstation answer), the request/answer mechanism must be designed as a critical section protected by a semaphore represented by logical variable *Lock* involved in the guards.

Obtaining a Solution.
An acceptable solution is obtained if a trace reaches a state where all tasks are in their final location "task assigned" exists. If such a trace exists, the recorded assignment parameters constitute the searched solution. Model-checking is a formal technique that explores the state space of a DES model to identify some properties, expressed using temporal logics, is enforced (or not) in the whole or partial state space. This technique can easily be used for reachability issue with a depth-first strategy to avoid explosion. The property can be expressed using CTL expression (Computation Tree Logic) [12] as: *EF("All tasks are assigned")* where E is the exist path quantifier, F the eventually temporal quantifier. This property means: there exists a path where "all the tasks are assigned" will be true one day.

4 Case Study: Trane's application

4.1 Case Study Description

This approach is evaluated using an industrial case study given by Trane Company. Trane is a firm selling cooling and heating air conditioning products and services. This firm's particularity is that the production is organized in manual assembly mixed model lines according to the DFT (Demand Flow Technology) basics. Because of the products' dimensions, a well-balanced assembly line is mandatory to avoid stocks between assembly process and its feeders. A well-balanced line must respect a production pace, the takt time.
The considered case study first three steps of a 14 operations assembly line are considered to be sure to finish a product's part before the test mandatory performed on the fourth workstation. The description and the duration of some tasks are given in Table 1 and the precedence graph is given on Figure 2. The targeted takt time is 67 minutes (4020 seconds). The initial line balancing is assumed to be known as depicted on the Figure 3 (maximum takt deviation = 228 seconds).

Table 1. Some Tasks desciption

Name	Description	Duration (minutes)	Duration (seconds)
1	Prépa & pose Base	3,3	198
2	Pose Evaporateur	6,4	384
15	Pose compresseurs C1	22,55	1353
16	Pose compresseurs C2	2255	1353
17	Brasage ligne Compresseur C1	16,2	972

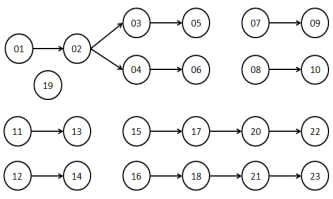


Fig. 2. Precedence Graph

4.2 Results

Example 1.

For the first example, the delay is detected during the task t_3 on the first workstation (this task is longer than planned, but it can't be moved, because it is already started). The two first tasks assigned to the first workstation are already finished. As a consequence, the remaining available capacity is reduced.

With a delay of 60 seconds a new assignment is found: the task t_9 could be moved from the first to the second workstation leading to the following balancing (Figure 4) (maximum takt deviation = 280 seconds, Fig.4).

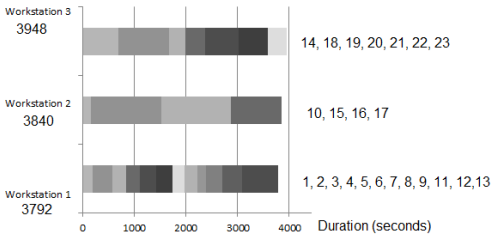


Fig. 3. Initial Line balancing

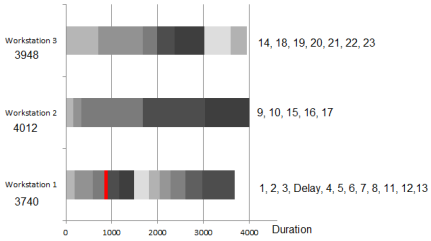


Fig. 4. First disturbance, 60s delay, corrected line balancing

Example 2.

For the second example, the delay (120 seconds) is detected during the task t_{15} on the second workstation. The 15 first tasks assigned to the two first workstations are already finished. According to these constraints no new better solution could be found (maximum takt deviation = 280 seconds). With a delay superior to 165 seconds, there is no acceptable solution agreeing with precedence constraints and the takt time constraint. (In this case, we must raise the takt time constraint value to obtain a solution.)

5 Conclusions, Future Works and Perspectives

In this article, we have shown how the reachability analysis could be used for an assembly line rebalancing. This algorithm is inserted in a predictive/reactive process in an intelligent manufacturing system context. It quickly gives an acceptable solution to adapt locally the predictive optimal balancing.

But sometimes, if there is no free space left, delays could not be absorbed by a simple reassignment of tasks. That is why our future works would deal with the parallelization of tasks. Furthermore, the cost of the reconfiguration would be included in the model. In fact, sometimes moving a task is more expensive than just dealing with the delay. Of course, this new approach will be compared with other methods concerning its ease of initialization, and the execution speed.

References

1. Becker, C., Scholl, A., 2006. A survey on problems and methods in generalized assembly line balancing. European Journal of Operational Research 168, 694–715. doi:10.1016/j.ejor.2004.07.023
2. Boysen, N., Fliedner, M., Scholl, A., 2008. Assembly line balancing: Which model to use when? International Journal of Production Economics 111, 509–528. doi:10.1016/j.ijpe.2007.02.026
3. Boysen, N., Fliedner, M., Scholl, A., 2007. A classification of assembly line balancing problems. European Journal of Operational Research 183, 674–693. doi:10.1016/j.ejor.2006.10.010
4. Battaïa, O., Dolgui, A., 2013. A taxonomy of line balancing problems and their solution-approaches. International Journal of Production Economics 142, 259–277. doi:10.1016/j.ijpe.2012.10.020
5. Falkenauer, E., 2005. Line balancing in the real world. Presented at the PLM'05 : international conference on product life cycle management, pp. 360–370.
6. Scholl, A., 1999. Balancing and sequencing of assembly lines (Publications of Darmstadt Technical University, Institute for Business Studies (BWL)). Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL).
7. Grangeon, N., Leclaire, P., Norre, S., 2011. Heuristics for the re-balancing of a vehicle assembly line. International Journal of Production Research 49, 6609–6628. doi:10.1080/00207543.2010.539025
8. Makssoud, F., Battaïa, O., Dolgui, A., 2013. Reconfiguration of Machining Transfer Lines, in: Borangiu, T., Thomas, A., Trentesaux, D. (Eds.), Service Orientation in Holonic and Multi Agent Manufacturing and Robotics, Studies in Computational Intelligence. Springer Berlin Heidelberg, pp. 339–353.
9. Estellon, B., Gardi, F., Nouioua, K., 2008. Two local search approaches for solving real-life car sequencing problems. European Journal of Operational Research 191, 928–944. doi:10.1016/j.ejor.2007.04.043
10. Behrmann, G., Brinksma, E., Hendriks, M., & Mader, A. (2005, April). Production scheduling by reachability analysis-a case study. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International* (pp. 140a-140a). IEEE.
11. Subbiah, S., Engell, S., 2010. Short-Term Scheduling of Multi-Product Batch Plants with Sequence-Dependent Changeovers Using Timed Automata Models, in: S. Pierucci and G. Buzzi Ferraris (Ed.), Computer Aided Chemical Engineering. Elsevier, pp. 1201–1206.
12. Clarke andClarke, E.M., Emerson, E.A., 1982. Design and synthesis of synchronization skeletons using branching time temporal logic, in: Kozen, D. (Ed.), Logics of Programs, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 52–71.
13. Lemattre, T., 2013. Allocation de fonctions de commande de systèmes critiques par recherche d'atteignabilité dans un réseau d'automates communicants. École normale supérieure de Cachan - ENS Cachan.Behrmann, G., Brinksma, E., Hendriks, M., Mader, A., 2005. Production Scheduling by Reachability Analysis - A Case Study, in: Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International. Presented at the Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International, p. 140a–140a. doi:10.1109/IPDPS.2005.363
14. Alur, R., Dill, D.L., 1994. A theory of timed automata. Theoretical Computer Science 126, 183–235. doi:10.1016/0304-3975(94)90010-8