Proc. Estonian Acad. Sci. Eng., 2002, **8**, 4, 211–222
https://doi.org/10.3176/eng.2002.4.01

Solving line balancing problems with model checking

Jüri Vain, Ingmar Randvee, Tiit Riismaa, and Juhan Ernits

Institute of Cybernetics at Tallinn Technical University, Akadeemia tee 21, 12618 Tallinn, Estonia; vain@ioc.ee

Received 14 June 2002

Abstract. A two-step technique for solving large-scale assembly line balancing problems is proposed. In the first step, an approximate global solution is found to the task assignment problem using the algorithmic branch-and-bound method. In the second step, the workstations with critical workload are selected and their load recalculated using local models of finer granularity. The workstation models in the second step are defined as parallel compositions of timed automata to which the parametric model checking procedure can be efficiently applied. An illustrative example is given.

Key words: flexible manufacturing, behavioural modelling, timed automata, line balancing, model checking.

1. INTRODUCTION

An extensive research in structuring methods of flexible manufacturing systems (FMS) has been carried out in recent years [1,2]. One basic group of problems to be repeatedly solved for production systems is related to the workflow structuring and line balancing. Solutions to these problems and efficiency of the applied methods depend on the granularity level of the model, constraints and implementation details to be taken into account. For instance, the SALOME technique as reported in [3,4] can arrange hundreds of tasks to the best logical structure of workstations (WS) using only task level estimated characteristics. On the other hand, the workstation time, i.e., the time needed to accomplish all tasks of the workstation, may depend on the given task assignment, machining modes, priorities, and other criteria. Thus the actual workstation time may differ from the simple sum of the roughly estimated task times.

In this paper, a two-step technique for solving assembly line balancing problems is proposed. In the first step, an approximate global solution to the task assignment problem is found using structural optimization algorithms [4–6]. In the detailed level, the model checking technique [7,8] is applied to distinct workstation models in order to get more accurate estimates of the workstation time. Since the global cycle time is defined by the maximum over workstation times, the whole production rate depends on the quality of this estimate. As a result of finer grain analysis, the optimal assignment of assembly line tasks, prescribed by the upper level solution, can be further improved, or on the contrary, denied if precise workstation time estimates exceed rough estimates of the cycle time.

It is supposed that each workstation model can be represented as a composition of timed automata [7,9,10], constructed on the basis of fine grain task models. Concentrating many implementation details into the workstation model has also some drawbacks considering the complexity of model checking. Model checking as a finite state assessment method suffers from inefficiency in case of a large number of parallel automata models. Still, since the efficiency depends on the balance between the size and the granularity of the models, efficient solution can generally be achieved in such cases when there are tens of tasks and few parallel machines in the workstation.

Our technique is demonstrated on the simple assembly line balancing problem (SALBP) using the benchmark data sets in the program SALOME-1 [11]. The construction of timed automata and the model checking procedure is illustrated using the workstation of an educational FMS. It is also assumed that the proposed modelling technique can be implemented for several types of FMS other than assembly lines [12–14].

The paper is organized as follows. First, in Section 2 the two-step modelling framework based on a SALBP is discussed. Section 3 exposes the main contribution of the paper, the methodology of constructing timed automata representing operation-level models. The model checking technique is briefly described in Section 4. An illustrative example of the operation-level model together with corresponding timed automata is exposed in Section 5. The paper ends with concluding remarks.

2. MODELLING FRAMEWORK FOR THE ASSEMBLY LINE BALANCING PROBLEM

For solving a SALBP by using the two-step technique we need two models. The first one, the line model (L-model), is given in terms of line tasks [3]. The second one, the operation-level model (O-model), describes every line task in terms of operations. An operation represents a subtask or its component up to the elementary activity of a machine. The line model is given by the list of tasks, task precedence graph, and estimated execution time for each task.

Classical SALBP is stated as follows: minimize the balance delay time, provided that the cycle time, task times, and the line model are given. The balance delay time characterizes capacity utilization of the line and is calculated as

$$BD = T - t_{\text{sum}},$$

where T is capacity supply of the line ($T = m/c$, m is number of workstations, and c is cycle time) and t_{sum} denotes the sum of task times.

The solution of SALBP consists of a minimal number of workstations, assignment of tasks to workstations, minimal balance delay time, and minimal cycle time (for given minimal number of workstations).

Considering the L-level problem, we rely on a bidirectional branch-and-bound procedure SALOME-1 [4] under the following general assumptions:

– any task of the line can be performed on any workstation;

– the task time does not depend on the number of tasks assigned to a station;

– any workstation can perform at most one task at a time;

– any task can be performed at most in one workstation at a time.

The O-model (tuned to the analysis of a single workstation load) is based on the precedence graph of operations, operation times, and syncronization constraints between concurrent tasks.

The O-level analysis problem is stated as follows: ensure that the workstation time is less or equal to the cycle time, provided the L-level task assignments and refined constraints on task execution are given.

The modelling assumptions of O-level analysis problems are usually stronger than those of the L-level analysis. For instance, the workstation can handle more than one operation at a time, the processing rate depends on the number of tasks assigned to the station, splitting of tasks may be allowed (in the case of parallel tasks), etc.

Evidently, if the workstation time of all stations with critical load can be reduced, then the cycle time can also be reduced when preserving optimal assignment. On the other hand, if the workstation time estimate exceeds the given cycle time, then the L-level analysis has to be repeated with new data based on the O-level analysis.

As it follows from optimal solutions for 282 assembly line balancing cases, using test data sets for the program SALOME-1 [11], only a few workstations with minimal and near-minimal idle times are to be checked. Also, the number of tasks in these workstations may not be very large. The number of tasks in a workstation (taken over 6019 stations) was distributed as follows: from 1 to 5 tasks – 74%, from 6 to 10 tasks – 22%, from 11 to 15 tasks – 3%.

As an illustrative example we consider the data set BOWMAN8.in2 with cycle time 20 s and precedence graph as depicted in Fig. 1.

The algorithm gives two optimal assignments with equal balance delay times.

Assignment 1: $\text{WS}_1(T_1, t^1_{\text{idl}} = 9\,\text{s})$, $\text{WS}_2(T_2, t^2_{\text{idl}} = 3\,\text{s})$, $\text{WS}_3(T_3, T_4, t^3_{\text{idl}} = 6\,\text{s})$, $\text{WS}_4(T_5, T_6, t^4_{\text{idl}} = 0)$, $\text{WS}_5(T_7, T_8, t^5_{\text{idl}} = 7\,\text{s})$.

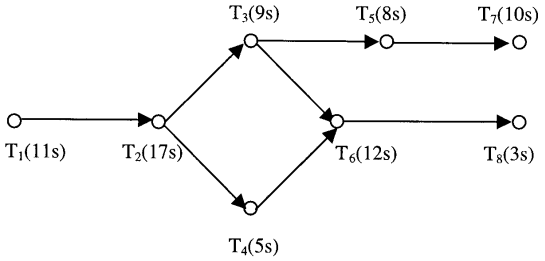

Fig. 1. Task precedence graph of the case study BOWMAN8.in2.

Assignment 2: $\mathrm{WS}_1(T_1, t^1_{\mathrm{idl}} = 9\,\mathrm{s})$, $\mathrm{WS}_2(T_2, t^2_{\mathrm{idl}} = 3\,\mathrm{s})$, $\mathrm{WS}_3(T_3, T_5, t^3_{\mathrm{idl}} = 3\,\mathrm{s})$, $\mathrm{WS}_4(T_4, T_6, T_8, t^4_{\mathrm{idl}} = 0)$, $\mathrm{WS}_5(T_7, t^5_{\mathrm{idl}} = 10\,\mathrm{s})$.

Here $\mathrm{WS}_i(T_j, \ldots T_k, t^i_{\mathrm{idl}})$ denotes assignment of tasks $T_j, \ldots T_k$ to the ith workstation with idle time t^i_{idl}. Comparing the idle time of stations it is easy to see that only station 4 in both cases ($\mathrm{WS}_4(T_5, T_6, t^4_{\mathrm{idl}} = 0)$ and $\mathrm{WS}_4(T_4, T_6, T_8, t^4_{\mathrm{idl}} = 0)$ must be studied in detail by the O-level analysis.

3. CONSTRUCTING O-LEVEL TIMED AUTOMATA

This section describes the construction of operation-level workstation models that are compositions of detailed task models, and a workstation configuration model.

By the ith workstation's WS_i configuration model $\boldsymbol{M}^i$ we mean the composition of machine models $\boldsymbol{M}^i = \|_l \boldsymbol{M}_l$, where each $\boldsymbol{M}_l$ performs the tasks sequentially ($\|_l$ – denotes parallel composition of models). Machine models are synchronized through processes $\boldsymbol{P}^i$ that represent the result of certain task scheduling strategy. (To avoid modelling process schedulers we assume for simplicity that processes are given by a L-level solution. For more details of modelling schedulers we refer to [10]).

1. Operation models. A machine M_l is characterized by a set of its operations $\boldsymbol{Op}^l$ and operational modes M^l. The attributes of an operation $op_j \in \boldsymbol{Op}^l$ may be priority, cost, time, pre- and postconditions, etc. For simplicity, we further consider only the execution time and cost. To construct a model $\boldsymbol{M}_l$ of a machine M_l we use the following definitions:

– a set of states $S(\boldsymbol{M}_l) = \{s_j : j = [1, |\boldsymbol{Op}^l|]\} \cup \{s_{\mathrm{idle}}\}$ so that for each operation $op_j \in \boldsymbol{Op}^l$ there is a state s_j, and s_{idle} is a special state that models the idle state of the machine M_l;

– a set of transitions $T(\boldsymbol{M}_l) = \cup_{l=[1,|\boldsymbol{Op}l|]}\{(s_j, s_{\mathrm{idle}}), (s_{\mathrm{idle}}, s_j)\}$;

– the duration d_j of operation op_j is modelled using the state invariant $\mathrm{Inv}(s_j) \equiv cl \le d_j$ and the guard $G(s_j, s_{\mathrm{idle}}) \equiv cl = d_j$, where cl is a clock variable of the machine model $\boldsymbol{M}_l$:

– the cost of operation op_j is modelled as an assignment $Asgn(s_{\mathrm{idle}}, s_j) \equiv a\_cost := a\_cost + Cost(op_j)$, where a_cost denotes the accumulated cost of

performing the operation op_j. Alternatively a_cost may model common cost for all operations of the machine or even of the workstation. If the accumulated cost is limited by some value *Limit*, it is represented as an operation guard

$$G(s_{\text{idle}}, s_j) \equiv (a\_cost + Cost(op_j)) < Limit.$$

2. Operational modes. Generally, operations of a machine M_l may be grouped into modes $M^l \subseteq 2^{Opl}$ (Fig. 2). Being in the mode $M^l_k \in M^l$, the machine is able to perform only operations $op_i \in M^l_k$. To perform an operation $op_j \notin M^l_k$ the machine should switch over to the mode M^l_r ($op_j \in M^l_r$). Switching takes time and costs and may be constrained so that only specified switching sequences are legal. That needs extension of the machine model $\boldsymbol{M^l}$ by introducing a *mode switching fragment*. Assume that each kth mode M^l_k is modelled separately as described above and has its idle state s^k_{idle}. Then the mode switching fragment consists of a set of transitions between the idle states s^k_{idle} and states that model switching operations (*switch* states in Fig. 2) exactly in the same way as other machining operations (see step 1 above).

3. Synchronizing processes and machine operations. The workstation processes $\boldsymbol{P^i}$ define the ordering of tasks. Each task is implemented as a sequence of operations. The workstation planner $\boldsymbol{A^i}$ makes planning, choosing appropriate operation sequences to execute the tasks on the given workstation configuration. Machine operations define the proper timing of operation sequences. Therefore, to model the cooperative behaviour of processes, planner and machines, initiations and terminations of tasks and individual operations must be synchronized.

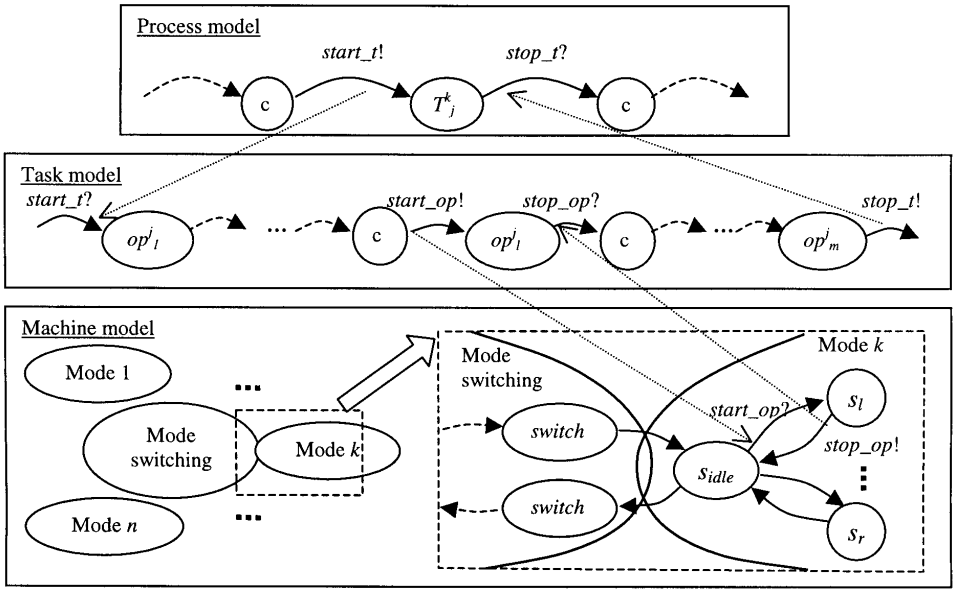

Fig. 2. Synchronization of workstation component models.

Synchronization is modelled using two types of synchronizing channels: *start* and *stop* between transitions of the process, task, and machine automata. Channel *start* synchronizes initiation and channel *stop* the completion of the task and operation executions in the models of processes, tasks, and machines. *Start* channels are directed from process models to task models and further from task models to machine models. *Stop* channels, on the contrary, are directed from the machine to the task and from the task to process models. Figure 2 depicts the synchronization mechanism between models of different levels.

4. MODEL CHECKING OF TIMED AUTOMATA

By model checking we generally mean an algorithmic method by which a desired behavioural property of a system is verified over the model through exhaustive enumeration of all states, reachables by the system, and the behaviours that traverse through them [8,15]. We formulate the model checking problem as a satisfiability problem $\boldsymbol{M} \models_? \varphi$, where φ denotes a behavioural property to be checked and $\boldsymbol{M}$ the model representing the behaviour to be checked.

The properties that the model must satisfy are given in timed modal logic L_s studied in [7] and used currently in the verifier of UPPAAL2k. We give a BNF-grammar of L_s:

$\varphi ::= \mathbf{A}\Box P_s \mid \mathbf{E}\Diamond P_s \mid \mathbf{E}\Box P_s \mid \mathbf{A}\Diamond P_s$
$P_s ::= AP \mid \neg P_s \mid (P_s) \mid P_s \vee P_s \mid P_s \wedge P_s \mid P_s \Rightarrow P_s$
AP ::= $Id_1.Id_2$ | CGuard | *I*Guard
CGuard ::= $Id \sim n \mid Id \sim Id \mid Id \sim Id + n \mid Id \sim Id - n$, where $n \in \mathbf{N}$
IGuard ::= $IExpr \sim IExpr \mid IExpr \neq IExpr$
IExpr ::= *Id* | *Id*[*IExpr*] | *n* | -*IExpr* | (*IExpr*) | *IExpr Op IExpr*
$\sim ::= < \mid \leq \mid \geq \mid > \mid =$
Op ::= + | – | * | /,

where P_s is a state formula, *AP* is the atomic state formula, *CGuard* and *IGuard* are the guards over clocks and integer variables, respectively, *Id* is the identifier name, $Id_1.Id_2$ is an identifier in the form "automaton_name.state_name", *n* is a natural number (including 0), and temporal modalities are: **A** – always, **E** – sometimes, $\Box$ – globally, $\Diamond$ – eventually.

Given a timed transition system $S = \langle S, s_0, R, L \rangle$, described by a network of timed automata, the L_s formulas are interpreted in terms of an extended state $s = \langle s, u \rangle$ where $s \in S$ is a state of a timed transition system and u is an assignment to formula clocks K; $R \subseteq S \times S$ is a precedence relation; $L: S \rightarrow 2^{AP}$ is a labeling function that assigns to each state a set of atomic propositions holding in that state.

A path on a transition system S is a (possibly infinite) sequence of states $\pi = s_0, s_1, ...$, where $\forall i \geq 0, (s_i, s_{i+1}) \in R$; π^i is suffix of the path π that begins from the ith state s_i. Let φ be a formula to be model checked. The satisfaction

relation |= between extended states (paths) and formulas is defined as the largest relation satisfying the following equivalences:

1) $S,s \models p \quad \Leftrightarrow \quad p \in L(s)$
2) $S,s \models \neg\varphi \quad \Leftrightarrow \quad S,s \not\models \varphi$
3) $S,s \models \varphi_1 \vee \varphi_2 \quad \Leftrightarrow \quad S,s \models \varphi_1$ *or* $S,s \models \varphi_2$
4) $S,s \models \varphi_1 \wedge \varphi_2 \quad \Leftrightarrow \quad S,s \models \varphi_1$ *and* $S,s \models \varphi_2$
5) $S,s \models \mathbf{E}\ \varphi \quad \Leftrightarrow$ there is a path π from state s, so that $S,\pi \models \varphi$
6) $S,s \models \mathbf{A}\ \varphi \quad \Leftrightarrow$ for all paths π from state s $S,\pi \models \varphi$
7) $S,\pi \models \varphi \quad \Leftrightarrow$ s is the first state of π and $S,s \models \varphi$
8) $S,\pi \models \neg\varphi \quad \Leftrightarrow \quad S,\pi \not\models \varphi$
9) $S,\pi \models \varphi_1 \vee \varphi_2 \quad \Leftrightarrow \quad S,\pi \models \varphi_1$ *or* $S,\pi \models \varphi_2$
10) $S,\pi \models \varphi_1 \wedge \varphi_2 \quad \Leftrightarrow \quad S,\pi \models \varphi_1$ *and* $S,\pi \models \varphi_2$
11) $S,\pi \models \Diamond\ \varphi \quad \Leftrightarrow$ exists $k \geq 0$, so that $S,\pi^k \models \varphi$
12) $S,\pi \models \Box\ \varphi \quad \Leftrightarrow$ for all $i \geq 0$ $S,\pi^i \models \varphi$

We write $S,s \models \varphi$ to express that the formula φ holds in a state s of a transition system S, and $S,\pi \models \varphi$ to express that the formula φ holds on a path π of S.

For example, the formula $A \Box (v_1 < v_2)$ says that invariantly $v_1 < v_2$ holds, and the formula $E\Diamond(A_1.s_i \wedge A_2.s_i)$ is true if the system can reach a global state where both automata A_1 and A_2 are in their states s_i.

5. MODELLING EXAMPLE

As an illustrative example, consider the educational FMS as a workstation with critical load to which five tasks are assigned. The IDEF0 diagram of that WS is given in Fig. 3.

The workstation process consists of the following sequence of tasks prescribed by the L-level precedence graph. The blank is placed on the conveyor1 (task 1). The conveyor1 transports the blank to the robot Mentor (task 2) that picks it from conveyor1 and places to conveyor2 (task 3). On the conveyor2 the blank is measured, classified and transported to a certain position (task 4) where the Serpent robot and CNC mill work it into the finished part (task 5). Task 5 in detail is: robot Serpent takes the good blank and places it to the CNC Mill worktable (operations 1–5, mode 2) where the blank is processed by one of four programs (modes 1–4). After completion of milling, the robot Serpent removes the item from CNC Mill and places it into the box (operations 1–5, mode 3) that is positioned under Serpent by the indexing table. If the new arrived blank is not good, robot Serpent removes it from the conveyor2 (operations 1–5 of mode 1). The indexing table turns to the right position after the blank is picked from the conveyor2. Only the initiation of this task is synchronized with the milling task since milling takes always longer time and continuation of the process depends on the latter.

The O-level model of the workstation is composed of separate task models according to the rules given in Section 3. As an example, the model of the task 5 with the process scheduler is depicted in Fig. 4.

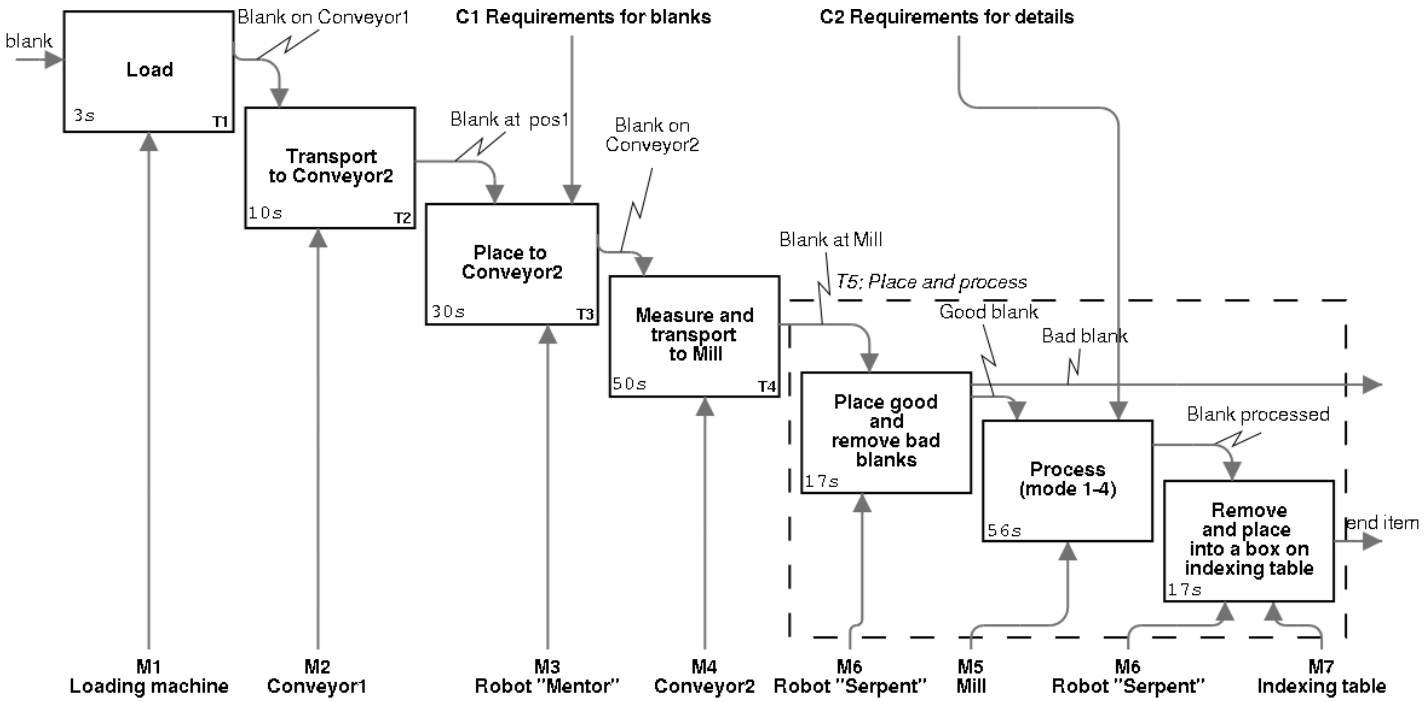

Fig. 3. IDEF0 representation of the load-critical workstation.

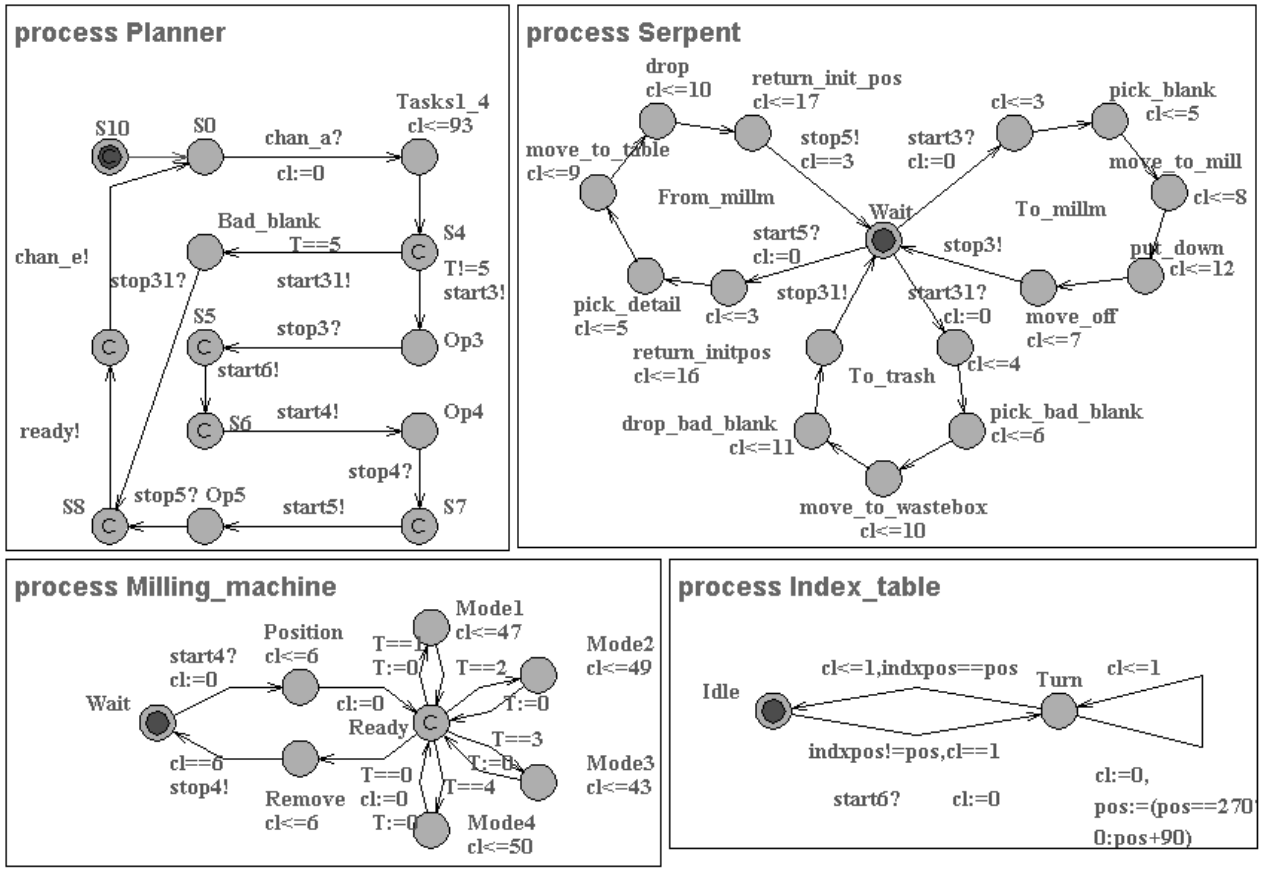

Fig. 4. Timed automata model of the task 5.

The model checking task to be solved to achieve more exact time estimates in comparison with the L-model can be expressed now using a L_s formula $A\Box(\text{PLANNER.CL} \leq \text{WSTIME})$, where PLANNER.CL denotes a clock in the automaton "Planner" that is reset each time when the processing of a detail in the workstation is completed, and WSTIME denotes the hypothetical workstation time towards which the O-model time estimate is checked. The initial value of WSTIME is naturally the L-level workshop time.

The results of checking the workstation time show that it is possible to reduce the cycle time from 200 to164–168 s depending on the milling mode, meanwhile preserving the L-level optimal solution of the assembly line structure. The reduction is mainly due to the partial overlapping of task operations inside the station. In case of the task 5, milling, indexing table, and bad blank removing operations can be performed in parallel.

6. CONCLUDING REMARKS

A typical line balancing problem involves hundreds of tasks. Due to the complexity of analysis, the used model and the set of modelling assumptions must be of a high level of abstraction. At the same time, separate workstations with prescribed task assignments can be easily modelled at a detailed level of granularity and under more sophisticated assumptions.

In this paper, a two-step technique for solving assembly line balancing problems as well as other resource assignment problems for FMS was introduced. It was shown that the technique combines the advantages of efficient coarse level algorithms and fine-grain model checking procedures.

The main result of the paper is the parametric model checking methodology that guides model construction and property specification for estimating workstation load and timing parameters in the presence of operational and timing constraints. The first step of the technique is demonstrated on the classical SALBP example using the benchmark test data sets for the line balancing program SALOME-1. The second step, construction of operation-level timed automata and the model checking procedure, was illustrated on the representative workstation of an educational FMS.

ACKNOWLEDGEMENTS

This work was partly supported by the Estonian Science Foundation under grants Nos. 4156 and 5086. The authors would like to acknowledge Prof. A. Scholl from the Darmstadt Technical University for SALBP-1 data sets, and Prof. R. Küttner and T. Otto from the Tallinn Technical University for the data of FMS workcell.

REFERENCES

1. Berio, G. and Vernadat F. B. New developments in enterprise modelling using CIM OSA. *Comput. Ind.*, 1999, **40**, 99–114.
2. Wieringa, R. A surwey of structured and object-oriented software specification methods and techniques. *ACM Comput. Surveys*, 1998, **30**, 459–527.
3. Scholl, A. *Balancing and Sequencing of Assembly Lines.* Physica-Verlag, Heidelberg, 1999.
4. Scholl, A. and Klein, R. SALOME: A bidirectional branch and bound procedure for assembly line balancing. *INFORMS J. Comput.*, 1997, **9**, 319–334.
5. Hoffmann, T. R. EUREKA: A hybrid system for assembly line balancing. *Manag. Sci.*, 1992, **38**, 39–47.
6. Johnson, R. V. Optimally balancing large assembly lines with FABLE. *Manag. Sci.*, 1988, **34**, 240–253.
7. Larsen, K., Pettersson, P., and Yi, W. UPPAAL in a nutschell. *Int. J. Softw. Tools Technol. Transf.*, 1997, **1**, 134–152.
8. Alur, R. and Dill, D. Automata for modelling real-time systems. *Theor. Comput. Sci.*, 1994, **126**, 183–236.
9. Hune, T., Larsen, K., and Pettersson, P. Guided synthesis of control programs using UPPAAL. *Nordic J. Comput.*, 2001, **8**, 43–64.
10. Vain, J. and Küttner, R. Model checking – a new challenge for design of complex computer-controlled systems. In *Proc. 5th International Conference on Engineering Design and Automation.* Las Vegas, 2001 (Parsaei, H. R., Gen, M., Leep, H. R., and Wong, J. P., eds.). CRC Press, CD-ROM, 593–598.
11. Scholl, A. and Klein, R. Assembly line balancing. Technical University of Darmstadt, 1996. http://www.bwl.tu-darmstadt.de/bwl3/forsch/projekte/alb/.
12. Lawler, E. I., Lenstra, J. K., and Rinnooy Kan, A. H. G. Recent developments in deterministic sequencing and scheduling: A survey. In *Deterministic and Stochastic Scheduling.* NATO Advanced Study Institutes Series. Series C: *Mathematical and Physical Sciences* (Dempster, M. A. H., Lenstra, J. K., and Rinnooy Kan, A. H. G., eds.). Reidel, 1982, 35–73.
13. Randvee, I., Riismaa, T., and Vain, J. Optimization of holonic structures. In *Proc. Workshop on Production Planning and Control WPPC'2000*. Mons, 2000. Ateliers de la FUCaM, Mons, 2000, 61–66.
14. Littover, M., Randvee, I., Riismaa, T., and Vain, J. Optimization of the structure of multi-parameter multi-level selection. In *Proc. 17th International Conference on CAD/CAM, Robotics and Factories of the Future, CARS&FOF 2001*. Durban, 2001 (Bright, G. and Janssens, W., eds.). University of Natal, Durban, 2001, 317–322.
15. Clarke, E. M., Grumberg, O., and Peled, D. A. *Model Checking*. MIT Press, Cambridge, MA, 1999.

Mudelkontrolli kasutamine koosteliini tasakaalustusülesande lahendamisel

Jüri Vain, Ingmar Randvee, Tiit Riismaa ja Juhan Ernits

Tüüpiline koosteliini tasakaalustusülesanne haarab sadu toiminguid. Et ülesanne oleks praktiliselt lahendatav, esitatakse nii kasutatav mudel kui ka piiravad tingimused tugevasti üldistatud kujul. Teiselt poolt on võimalik tööjaamade tegelikku keerukust paremini arvestada, kui kasutada tasakaalustusülesande lahendina

leitud toimingute jaotust üksikute tööjaamade vahel koos koormuskriitiliste tööjaamade täpsustatud mudelitega.

Käesolevas töös on esitatud kaheetapiline lahendusskeem toimingute optimaalseks jaotamiseks tööjaamade vahel. See ühildab tuntud tasakaalustusmeetodite ja üksiku tööjaama detailse mudelkontrolli eelised. Esitatu uudsus seisneb mudelkontrolli metodoloogia integreerimises koosteliini tasakaalustusülesande lahendusskeemi. Meetodi demonstreerimiseks on kasutatud Tallinna Tehnikaülikooli paindtootmissõlme maketti, mis on kirjeldatud ajaga automaatide formalismis, ning koosteliinide tasakaalustusülesande lahendusmeetodite testimiseks koostatud avalikku lähteandmete kogu.