Simulation-optimization of schedules with sequence dependent setup times based on Petri net models

Gašper Mušič[(a)]

[(a)]University of Ljubljana, Faculty of Electrical Engineering, Ljubljana, Slovenia

[(a)]gasper.music@fe.uni-lj.si

ABSTRACT

The paper deals with simulation-optimization of schedules with sequence dependent setup times. Such scheduling problems are commonly found in industry where machine setups or cleaning procedures are required among different jobs. The problems can be effectively modelled by simple Coloured Petri nets (CPNs). CPN modelling is combined with new type of simulation based on predefined transition set sequence conflict resolution strategy and standard permutation representation of job schedules. This enables generation of neighbouring solutions that are always feasible and standard local search optimization algorithms can be effectively applied to derived models. Modelling approach and neighbourhood construction procedure are explained in detail. Results of tests on sample scheduling problems by simulated annealing and genetic algorithms are provided.

Keywords: Petri nets, simulation, optimization, scheduling, local search

1. INTRODUCTION

Scheduling problems have been one of the classical research topics within Operations research for decades. Recently, manufacturing operations scheduling is gaining importance in line with wide informatization of industrial processes that should facilitate the optimisation of various operational aspects of such systems. Among those, efficient operations scheduling can contribute to better production responsiveness by more efficient processing of released manufacturing jobs.

Petri nets (PNs) are a modelling technique that is well suited to description of manufacturing systems. In contrast to most system description languages, PNs are state and action oriented at the same time providing an explicit description of both the states and the actions. When modelling manufacturing systems PNs have several advantages, such as efficient modelling of concurrency and synchronization, ability of capturing functional, temporal and resource constraints within a single formalism (Recalde et al. 2004), and simple representation of production systems' specific properties (Tuncel and Bayhan 2007). These can be supplemented by various extensions that enable quantitative performance analysis of modelled systems. A rich variety of extensions of the original PN formalism include the time concept, considering both deterministic and stochastic time (Viswanadham and Narahari 1992). In this way, PNs can be applied for simulation-based operations scheduling, that is, timed simulation can be performed to evaluate the quality of the schedules under certain operating conditions (Piera and Mušič 2011). If the schedule is modelled properly, the job completion times can be read from the timestamps of tokens in the final marking obtained by PN simulation. As the objective to be minimized is always a function of the completion times of the jobs (Pinedo 2008), this enables to implement scheduling systems with various objectives.

In our previous work a simulation based optimization approach applying PNs was intensively studied, as well as other, more classical approaches, such as dispatching rules and reachability tree based heuristic search (Gradišar and Mušič 2007, Löscher et al. 2007). In particular, the investigations focused on combination of PN modelling approach and local search methods (Löscher et al. 2007, Mušič 2012b). Other authors focus on the use of PN models for deadlock-free design, resource allocation and scheduling optimization (Sindičić et al. 2012, Huang et al. 2013, Xing et al. 2012), and use of PNs with other metaheuristics, e.g. genetic algorithms (Mejía et al. 2012, Han et al. 2014).

This paper focuses on PN models of scheduling problems with sequence dependent setup times (Kurz and Askin 2004) and explores the possibilities of PN models use in conjunction with state-of-the-art local search algorithms. The previous works on standard flow-shop and job-shop problems are generalized, so that also less restricted problem classes of larger practical relevance can be addressed and PN based scheduling methods are brought closer to the practical needs in manufacturing and services.

2. PETRI NET MODELLING OF SCHEDULES WITH SEQUENCE DEPENDENT SETUP TIMES

Petri net models of scheduling problems have been described in our previous works (Mušič 2011, 2012b, 2015). Both Place/Transition and Coloured Petri net classes have been used for that purpose. Yet, schedules with sequence dependent setup times have not been addressed so far.

Such problems are of great practical relevance as

complex flow lines exhibiting this property can be found in many industries. Ríos-Mercado and Bard (1998) state that sequence-dependent setup times are found in the container manufacturing industry as well as the printed circuit board industry. Also semiconductor and automobile industries encounter sequence dependent setup times, which result in difficult scheduling problems (Kurz and Askin 2004).

Here we follow the formulation of Ríos-Mercado and Bard (1998), who assume that setup can only be performed after the machine is no longer processing any job and the job for which setup is being performed is ready. The basic property that has to be captured within the model of such a schedule with sequence dependent setup times is the dependence between setup durations and job order. In the proposed model this is achieved by explicitly representing recently processed job by an additional control place. When using Place/Transition PNs a set of places is required for that purpose to distinguish among different jobs. A sample resulting model for a single machine with three jobs is shown in Figure 1.

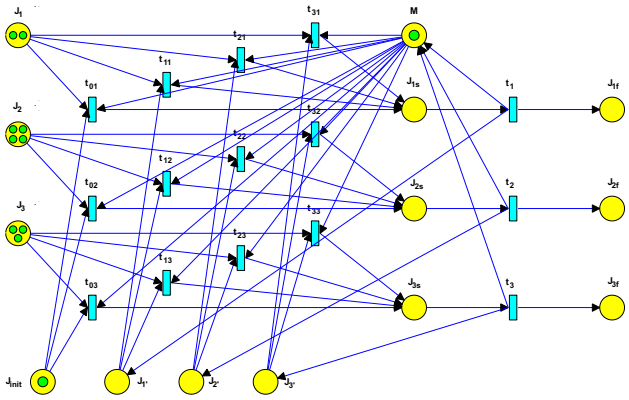

Figure 1: PN model of a sample scheduling problem with sequence dependent setup times

Here setup times are assigned to transitions t_{ij}, where i and j denote the previous and the current job in the sequence, respectively. Marking of J_i denotes i-th job requirements, while $J_{i'}$ is used to memorize the previous job and choose correct transition with proper setup time for the next firing. As initially all $J_{i'}$ are empty, J_{init} is used to enable an initial transition, the choice of which depends on the schedule. Similarly, also more than one t_{ij} can be enabled in subsequent firing steps, but all the transitions compete for a single shared resource M. The choice on which of them will fire and occupy the resource depends on the schedule. After the setup is complete the job operation is processed, which is modelled by transitions t_i with corresponding operation duration. Tokens collected in J_{if} indicate processed jobs. If more stages are required, these tokens are the input to a subsequent stage, so the model can be easily expanded to an arbitrary number of stages.

The number of nodes in the model is polynomial with the number of different jobs, so a graphical representation of the Place/Transition PN model quickly becomes unmanageable. On the other hand it possesses large degree of self-similarity, which indicates that representation can be

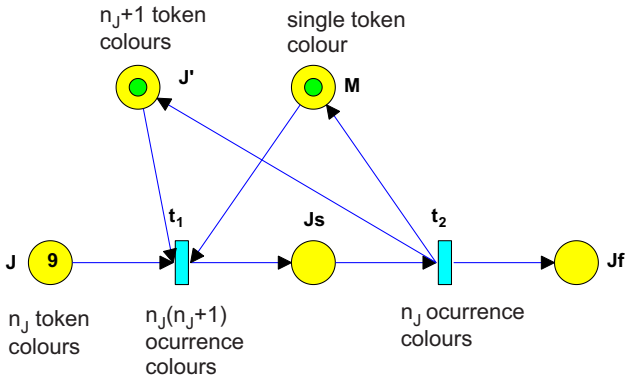

Figure 2: CPN model of a scheduling problem with sequence dependent setup times

simplified by employing Coloured Petri nets (CPNs). Figure 2 shows the same model in CPN framework.

Here all job types can be represented by a single node, containing tokens of various colours. Similarly, different setup times or processing times are represented by a single transition with multiple occurrence colours.

3. PETRI NET SIMULATION OF SCHEDULES WITH SEQUENCE DEPENDENT SETUP TIMES

To use the derived model within simulation-optimization approach, a simulation based evaluation of schedule objective function needs to be implemented. The simulation run must depend on externally changing influential variables, whose changes are driven by optimization algorithm.

To formalize the simulation algorithm, the modelling framework is first presented in a formal manner.

3.1. Coloured Petri nets

For clarity of presentation, a simplified version of Coloured Petri nets without arc expressions and guards will be considered in the following. Compared with standard CPNs (Jensen 1997), no arc inscriptions other than weight are used, and transition inscriptions are limited to specification of time delay only. This allows a more focused presentation of time aspects while the mechanism of time inclusion is kept as close as possible to the one of standard CPNs. This enables a straightforward extension of presented concepts to standard CPNs.

A formal definition of such a CPN is given as follows. Note that the definition closely follows one of the representations used in Basile et al. (2007) with an important difference: a different interpretation of transition delays is used, which is closer to that of Jensen (1997).

A $CPN = (\mathcal{N}, M_0)$ is a Coloured Petri net system, where: $\mathcal{N} = (P, T, Pre, Post, Cl, Co)$ is a Coloured Petri net structure:

- $P = \{p_1, p_2, \ldots, p_k\}, k > 0$ is a finite set of places.
- $T = \{t_1, t_2, \ldots, t_l\}, l > 0$ is a finite set of transitions (with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$).
- Cl is a set of colours.

– $Co : P \cup T \rightarrow Cl$ is a colour function defining place marking colours and transition occurrence colours. $\forall p \in P, Co(p) = \{a_{p,1}, a_{p,2}, \ldots, a_{p,u_p}\} \subseteq Cl$ is the set of u_p possible colours of tokens in p, and $\forall t \in T, Co(t) = \{b_{t,1}, b_{t,2}, \ldots, b_{t,v_t}\} \subseteq Cl$ is the set of v_t possible occurrence colours of t.

– $Pre(p,t) : Co(t) \rightarrow Co(p)_{MS}$ is an element of the pre-incidence function and is a mapping from the set of occurrence colours of t to a multiset over the set of colours of p, $\forall p \in P, \forall t \in T$. It can be represented by a matrix whose generic element $Pre(p,t)(i,j)$ is equal to the weight of the arc from p w.r.t colour $a_{p,i}$ to t w.r.t colour $b_{t,j}$. When there is no arc with respect to the given pair of nodes and colours, the element is 0.

– $Post(p,t) : Co(t) \rightarrow Co(p)_{MS}$ is an element of the post-incidence function, which defines weights of arcs from transitions to places with respect to colours.

$M(p) : Co(p) \rightarrow \mathbb{N}$ is the marking of place $p \in P$ and defines the number of tokens of a specified colour in the place for each possible token colour in p. Place marking can be represented as a multiset $M(p) \in Co(p)_{MS}$ and the net marking M can be represented as a $k \times 1$ vector of multisets $M(p)$. M_0 is the initial marking of a CPN.

Timed models

As described in Bowden (2000), there are three basic ways of representing time in PNs: firing durations (FD), holding durations (HD) and enabling durations (ED). When using HD principle, a firing has no duration but a created token is considered unavailable for the time assigned to transition that created the token.

HD principle is somewhat restrictive, as can be seen in a PN where a conflict appears. In this case more transitions are enabled by one marking, but not all can fire simultaneously. A choice has to be made on which of them will actually fire. With HD principle the firing occurs immediately after the choice is made and the conflict is resolved, and the delay is only active afterwards. This indicates that HD concept can only model non-preemptive tasks in the system. A more general concept is presented and formally described in Lakos and Petrucci (2007). When modelling several performance optimization problems, e.g. scheduling problems, a general framework is not needed. It is natural to use HD when modelling most scheduling processes as operations are considered non-preemptive. HD principle is also used in the timed version of CPNs defined by Jensen (1997), where the assignment of delays both to transitions and to output arcs is allowed. We further simplify this by allowing time delay inscriptions to transitions only. This is sufficient for the type of examples investigated here, and can be generalized if necessary.

To include a time attribute of the marking tokens, which implicitly defines their availability and unavailability, the notation of Jensen (1997) will be adopted. A group of coloured tokens is accompanied with a time stamp, which defines the time instant at which the tokens become available. The time refers to the observation time and is measured by a global clock. A timestamp is assigned to every token at the moment of token creation; this can be at the definition of the initial marking or at a transition firing. A group of tokens is available and can participate in transition enabling if and only if the assigned timestamp values are lower or equal to the global clock value. In the opposite case the tokens are declared unavailable. As the global clock is increasing a momentarily unavailable token can become available at some future point in time.

The timestamp is written next to the token number and colour and separated from the number by @. E.g., two c-coloured tokens with time stamp 10 are denoted 2‘c@10. A collection of tokens with different colours and/or time stamps is defined as a multiset, and written as a sum (union) of sets of timestamped tokens. E.g., two c-coloured tokens with timestamp 10 and three d-coloured tokens with timestamp 12 are written as 2‘c@10+3‘d@12.

Time stamps are elements of a time set TS, which is defined as a set of numeric values. In many software implementations the time values are integer, i.e. $TS = \mathbb{N}$, the same assumption will be used in the sequel. The timestamps could also be admitted to take any positive real value including 0, i.e. $TS = \mathbb{R}_0^+$, the restriction to integers merely simplifies the presentation of examples. Timed markings are represented as collections of time stamps and are multisets over TS: TS_{MS}. By using HD principle the formal representation of a Coloured Timed Petri net (CTPN) is defined as follows.

$CTPN = (\mathcal{N}, M_0)$ is a Coloured Timed Petri net system, where:

– $\mathcal{N} = (P, T, Pre, Post, Cl, Co, f)$ is a Coloured Time Petri net structure with $(P, T, Pre, Post, Cl, Co)$ as defined above.

– $f : Co(t) \rightarrow TS$ is the time function that assigns a non-negative deterministic time delay to every occurrence colour of transition $t \in T$.

– $M(p) : Co(p) \rightarrow TS_{MS}$ is the timed marking, M_0 is the initial marking of a timed PN.

Coloured Petri net firing rule

The firing rule of a CTPN is defined in accordance with the HD timing principle. Functions Pre and $Post$ define the weights of directed arcs, which are represented by arc inscriptions in the matrix form. In the case when the all the weights in the matrix are 0, the arc is omitted. Let ${}^\bullet t_b \subseteq P \times Cl$ denote the set of places and colours which are inputs to occurrence colour $b \in Co(t)$ of transition $t \in T$, i.e., there exists an arc from every $(p,a) \in {}^\bullet t$ to t with respect to colours $a \in Co(p)$ and $b \in Co(t)$.

To determine the availability and unavailability of tokens, two functions on the set of markings are defined. The set of markings is denoted by $\mathbb{M}$. Given a marking and a time $\tau_i \in TS$, $m : P \times \mathbb{M} \times TS \rightarrow Co(p)_{MS}$ defines the number of available coloured tokens, and $n : P \times \mathbb{M} \times TS \rightarrow Co(p)_{MS}$ defines the number of unavailable coloured tokens for each place of a CTPN at a given model time $\tau_i \in TS$. As mentioned before, the model time is represented by a global clock.

Two timed markings can be added (denoted $+_\tau$) in a similar way as multisets, i.e. by making a union of the corresponding multisets. The definition of subtraction is somewhat more problematic. To start with, a comparison operator is defined. Let M_1 and M_2 be markings of a place $p \in P$. By definition, $M_1 \geq_\tau M_2$ iff $m(p, M_1, \tau_i) \geq m(p, M_2, \tau_i), \forall \tau_i \in TS, \forall a \in Co(p)$.

Similarly, the subtraction is defined by the number of available tokens, and the subtrahend should not contain any unavailable tokens. Let M_1, M_2 and M_3 be markings of a place $p \in P$, $M_1 \geq_\tau M_2$, and $m(p, M_1, \tau_i)$, $m(p, M_2, \tau_i)$, and $m(p, M_3, \tau_i)$, be the corresponding numbers of available tokens at time τ_i, and $n(p, M_2, \tau_i) = 0$. The difference $M_3 = M_1 -_\tau M_2$ is then defined as any $M_3 \in \mathbb{M}$ having $m(p, M_3, \tau_i) = m(p, M_1, \tau_i) - m(p, M_2, \tau_i)$.

As the HD principle is used it is not important which of the available tokens is removed. All the subsequent transition firings only deal with present or future points in time, and all currently available tokens remain available for the eventual enablement of future firings. The implementation can be simplified if the new markings are always generated in the same way, e.g., by always removing the token with the most recent timestamp first.

Using the above definitions, the HD-based firing rule of a CTPN can be defined. Given a marked $CTPN = (\mathcal{N}, M)$, a transition t is *time enabled* at time $\tau_i \in TS$ w.r.t occurrence colour $b \in Co(t)$, denoted $M[t_b\rangle_{\tau_i}$ iff $m(p, M, \tau_i) \geq Pre(p,t)(b)$, $\forall p \in \bullet t_b$. An enabled transition can fire, and as a result removes tokens from input places and creates tokens in output places.

If transition t fires w.r.t occurrence colour b, then the new marking is given by

$$M'(p) = M(p) -_\tau Pre(p,t)(b)@\tau_i \\ +_\tau Post(p,t)(b)@(\tau_i + f(t,b)), \forall p \in P \quad (1)$$

If marking M_2 is reached from M_1 by firing t_b at time τ_i, this is denoted by $M_1[t_b\rangle_{\tau_i} M_2$. The set of markings of CTPN $\mathcal{N}$ reachable from M is denoted by $R(\mathcal{N}, M)$.

3.2. Simulation with predefined transition set sequences

Petri nets can be used either to constructively build a schedule, employing a given set of scheduling rules, or to evaluate the objective function for a schedule, given in advance. Within the simulation-optimization approach used in this paper we follow the second approach.

The basic problem that has to be resolved when evaluating PN represented schedules is to translate a particular schedule into on-the-fly conflict resolution strategy that will ensure a proper ordering of conflicting transitions. The conflicting transitions represent jobs competing for a shared resource at a given moment in simulation time.

The conflict resolution strategy is here embedded within the PN simulation algorithm and parameterized by a prescribed sequence of transition sets.

Given a set $T_{be} \subseteq T \times Cl$ of time enabled transition t occurrence colours (t, b), shortly denoted as t_b

$$T_{be} = \{t_b : t \in T, b \in Co(t), M[t_b\rangle_{\tau_i}\}$$

and a set of schedule relevant transition occurrence colours

$$T_{sch} = \{t_b : \exists b \in Co(t), Pre(r,t)(b) > 0\}$$

where $r \in P$ is a resource place, a *transition set sequence* S is considered

$$S = \{T_k : \\ T_k = \{t_{b,a} : t \in T, b \in Co(t), a \in Co(J), \\ Pre(J,t)(b,a) > 0\}\}$$

where $J \in P$ is the job place (see Fig.2), a is the token colour of k-th job in the scheduled sequence, and $k = 1 \ldots k_{max}$ is a sequence state.

We say that transition t is *sequence enabled* at sequence state s w.r.t occurrence colour $b \in Co(t)$, denoted $M[t_b\rangle_{T_k}$ iff $t_b \in T_k \vee t_b \notin T_{sch}$. Note that transitions $T \notin T_{sch}$ are always sequence enabled.

During simulation run, at the stage where firing transitions are determined, the transition enabling check within the simulation algorithm implements an additional filter, which filters out the transitions that are not sequence enabled. Only transitions that are both marking- and sequence-enabled are permitted to fire and thus change the PN marking. The new marking is computed according to Eq. (1). Additionally sequence state k is increased, whenever a schedule relevant transition is fired. Such a simulation model will be denoted $PN(S)$.

This way a link is established between the scheduled job sequence and PN transition firing sequence. Compared to our previous work, e.g. (Löscher et al. 2007, Mušič 2012b), the approach is generalized to permit more than one schedule relevant transition occurrence to fire at particular sequence state, which is achieved by specifying set sequence instead of simple firing sequence. A broader class of scheduling models can be addressed this way, such as the case of sequence dependent setup times, where the particular firing is chosen based on the previous system state.

Repeating the above described enablement checking step and marking calculation step evolves the PN model from initial marking, indicating the required number of individual jobs' repetitions to the final marking, indicating the number of finished jobs. Timestamps adjoined to tokens in the final marking indicate the job completion times, which permits to calculate an arbitrary schedule objective (Mušič 2012a, Napalkova et al. 2014).

4. PETRI NET BASED OPTIMIZATION OF SCHEDULES WITH SEQUENCE DEPENDENT SETUP TIMES

4.1. Local search

The proposed optimization approach is to use local search methods. Basic optimization algorithm is shown in Algorithm 1.

One of the central issue within the approach is the neighbour solution construction procedure. In Mušič (2011) a procedure has been proposed, which constructs a new scheduling solution by permuting sequence vectors related to transitions in the PN model that are linked to shared resources. The resulting solution is always feasible but the approach only works for standard job-shop and

Algorithm 1 Local Search

```
generate initial solution s;
i := 0;
repeat
  generate neighbour solution s' ∈ N(s);
  perform acceptance test of s';
  if test positive then
    s := s';
  end if
  i := i + 1;
until termination criterion
```

flow-shop scheduling problems. A more general construction procedure is presented in Mušič (2015), which can also be used for flexible job shop problems.

Alternatively, standard permutation representations of schedules can be employed (Cheng et al. 1996, Werner 2013). Among others, the *operation-based representation* is used, where job sequence is coded by a vector of integer indices. Job operations are represented by repetition of indices within the vector, where i-th repetition indicates the execution of i-th operation within the job. Such a representation is used in Xing et al. (2012) where schedules are evaluated by a custom algorithm that takes into account the transition precedence relations within the PN model and corresponding operation durations. Although the work of Xing et al. (2012) is focussed on maintaining neighbour solution feasibility, in particular in the case of deadlock-prone schedule models, the schedule evaluation algorithm only works effectively for particular problem classes. As such the approach is not directly applicable to problems with sequence dependent setup times.

Using the simulation approach with predefined transition set sequences, presented in the previous section, a new schedule evaluation procedure within the local search strategy is proposed here in this paper.

The schedule is represented by a permutation vector

$$F = \{\pi(1), \pi(2), \ldots, \pi(k_{max})\}$$

$f_k = \pi(k)$ denotes an integer obtained by permutation of $\{1, \ldots, k_{max}\}$ and k_{max} corresponds to

$$k_{max} = n_{stages} \cdot \sum_a M(J)$$

i.e., it represents the number of all required job operations.

At every optimization step, a change in the permutation vector is imposed, e.g. two randomly chosen elements are swapped:

$$N : F \rightarrow F'$$

$$F' = \{f_1, \ldots, f_{j-1}, f_k, \ldots, f_{k-1}, f_j, \ldots, f_{k_{max}}\}$$

Next the resulting vector is decoded into a sequence of transition sets. As the setup times are sequence dependent, no exact calculation of the setup duration is performed at this stage. A set of corresponding transitions that can trigger the setup is specified instead. This substantially simplifies the decoding step.

Next, the simulation is run with the determined transition set sequence as a parameter. During each simulation step, the compliance with the set sequence is maintained and in case of more than one transition in a sequence element, triggering of one of participating transitions is allowed, as shown above. This way, the actual setup time that applies in a given moment is implicitly defined by a combination of the sequence element and the schedule model. It is determined automatically by checking marking- and sequence- enablement of transitions within the PN model.

Finally, the simulation result is used to evaluate the objective function and the new schedule is kept or discarded, depending on the detailed implementation of the local search strategy.

The optimization algorithm is summarized as follows

Algorithm 2 Schedule optimization

```
generate PN model;
calculate the number of job operations;
generate permutation vector F;
decode initial set sequence S;
simulate PN(S);
calculate initial schedule and related objective function
value;
i := 0;
repeat
  generate neighbour solution F' ∈ N(F);
  decode set sequence S';
  simulate PN(S');
  calculate new schedule and related objective function
  value;
  perform acceptance test of F';
  if test positive then
    F := F';
  end if
  i := i + 1;
until termination criterion
```

By introducing eventual restrictions within neighbour solution construction procedure and by using different acceptance tests, various local search algorithms can be implemented, e.g., Simulated Annealing (SA) or Tabu search (Brucker 2007).

4.2. Genetic algorithms

The local search procedure can be easily adapted to the use of genetic algorithms (Cheng et al. 1996, Eiben and Smith 2015). The above described permutation vector is used as a chromosome, and the described vector change, which represents the neighbourhood operator in local search strategy, is used in mutation step of the genetic algorithm.

What remains is to define a crossover operator. Standard crossover operators for the permutation representations can be used, such as order crossover (OX), linear order crossover (LOX) or position-based crossover (PBX) (Cheng et al. 1999, Werner 2013, Eiben and Smith 2015).

Simulation approach with predefined transition set sequences is then used in the fitness evaluation stage of the genetic algorithm.

5. EXPERIMENTAL RESULTS

5.1. Single machine problem

The introductory example of Fig. 1 is used to show the feasibility of the approach. The setup times are integers selected from a uniform distribution between 2 and 4, and integer processing times are selected from a uniform distribution between 8 and 12.

Initial optimization was attempted by SA algorithm. Three problem instances were generated and 10 optimization runs were performed by each instance. Objective function was the makespan, i.e. the finish time of the last job. Figure 3 shows how objective function of the current solution changes with the iterations of the SA algorithm for the three problem instances. Figure 4 shows the convergence of the solutions for all optimization runs. It can be seen that although the convergence rates are different, all the runs converge to the same values for the same problem instances.

One of the solutions is illustrated by Gantt chart in Figure 5, where processing times for jobs J_1, J_2 and J_3

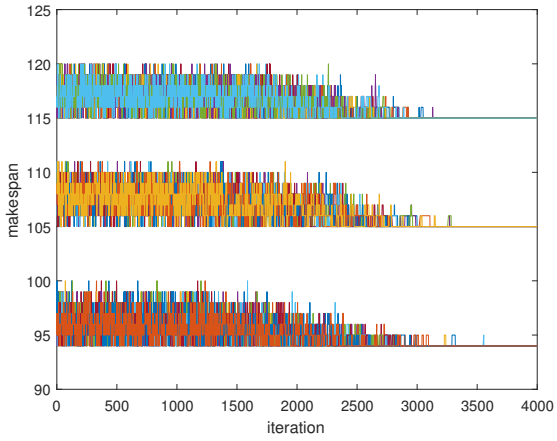

Figure 3: Solution changes during the run of SA algorithm

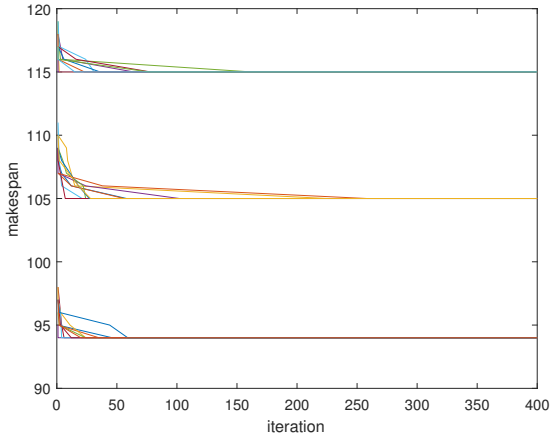

Figure 4: Convergence of solutions for three problem instances

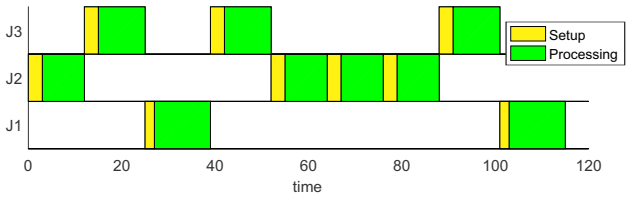

Figure 5: Sample Gantt chart for a problem with sequence dependent setup times

are 12, 9 and 10, respectively, initial setup times are 3, 3 and 4, and sequence depended setup times are given by matrix:

$$S_{ij} = \begin{bmatrix} 3 & 4 & 3 \\ 2 & 3 & 3 \\ 2 & 3 & 4 \end{bmatrix}$$

where S_{ij} denotes the setup time when job J_i is followed by J_j. The optimal makespan is 115 time units.

5.2. Two stage flow line

The two stage flow line example is inspired by the set of sample problems in Kurz and Askin (2004). Basically, the model of the first example is extended to 6 different jobs, and then repeated twice in sequel, where final job places of stage 1 are used as initial job places of stage 2. Integer setup times are selected from a uniform distribution between 5 and 90, and integer processing times are selected from a uniform distribution between 50 and 70.

Figure 6 illustrates a sample optimization run for the second example.

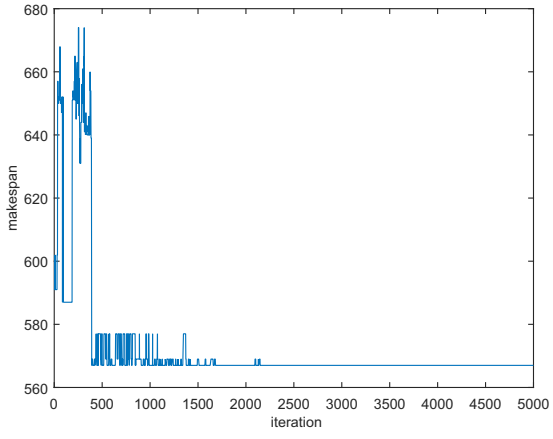

Figure 6: Solution changes during the run of SA algorithm for the two stage example

Additionally, the second example was optimized by genetic algorithm. The Matlab implementation of genetic algorithm within Global Optimization Toolbox was used, with custom defined fitness function, which implements the proposed set sequence simulation based evaluation of solutions. Additionally, custom crossover and mutation functions were defined, which are able to process permutation based solutions.

Although good solutions were obtained in general, significant dependence on algorithm parameter settings was observed, e.g., dependence on Fitness Scaling Options and Selection Options. In general, the settings that maintain greater population diversity lead to better overall results. Figures 7 and 8 show the convergence for one of the problem instances and two different sets of algorithm settings. The same crossover and mutation operators were used, but different fitness scaling (rank vs. proportional) and different selection settings (stochastic uniform vs. uniform).

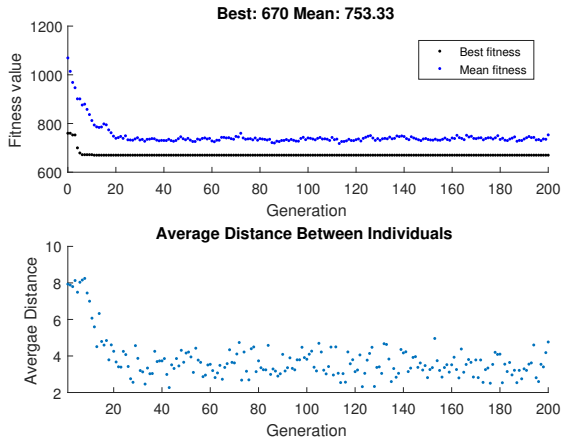

Figure 7: Optimization of the two stage example by genetic algorithm - fast convergence, poor solution

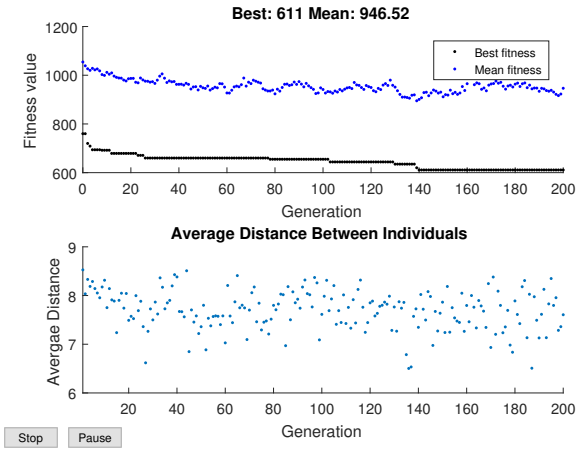

Figure 8: Optimization of the two stage example by genetic algorithm - slow convergence, good solution

Figures 9 and 10 illustrate the optimal solution by Gantt charts. Note that setup and processing stages of a job are represented by the same colour.

Finally, a comparison of different crossover operators was performed. No significant impact of the crossover operator choice to the solution quality is observed, which is illustrated in Figure 11 by shoving average resulting

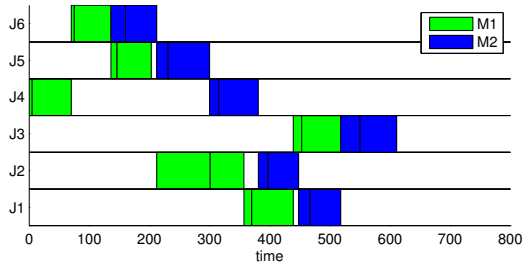

Figure 9: Gantt chart for the two stage example - job ordering

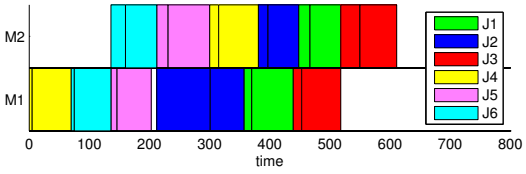

Figure 10: Gantt chart for the two stage example - machine usage

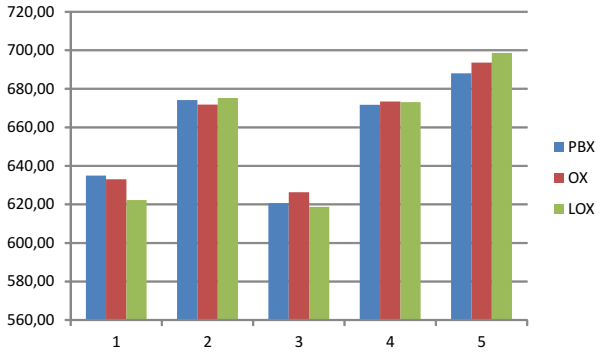

Figure 11: Average results of 10 optimization runs for five problem instances using different crossover operators

makespan of 10 optimization runs for five problem instances using operators PBX, OX, and LOX, respectively.

6. CONCLUSIONS

The paper shows how existing simulation-optimization framework using Petri net models and specific, sequence dependent PN simulation can be extended to the class of scheduling problems with sequence dependent setup times. Related PN modelling approach is presented, the advantage of Coloured Petri nets use is illustrated, and sequence dependent simulation algorithm is extended by allowing set sequence specification. This way an additional degree of freedom is introduced in specifying firing sequences, which permits to partially determine the actual sequence on-the-fly during simulation execution. Some illustrating examples show the feasibility of the approach in combination with local search techniques and genetic algorithms.

REFERENCES

Basile, F., Carbone, C. and Chiacchio, P., 2007. Simulation and analysis of discrete-event control systems based on Petri nets using PNetLab, *Control Engineering Practice*, 15, 241–259.

Bowden, F. D. J., 2000. A brief survey and synthesis of the roles of time in Petri nets, *Mathematical & Computer Modelling*, 31, 55–68.

Brucker, P., 2007. *Scheduling Algorithms*, 5th edn, Springer.

Cheng, R., Gen, M. and Tsujimura, Y., 1996. A tutorial survey of job-shop scheduling problems using genetic algorithms–I. representation, *Computers & Industrial Engineering*, 30 (4), 983–997.

Cheng, R., Gen, M. and Tsujimura, Y., 1999. A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies, *Computers & Ind. Engineering*, 36 (2), 343–364.

Eiben, A. E. and Smith, J. E., 2015. *Introduction to Evolutionary Computing*, Natural computing series, 2 edn, Springer-Verlag, Berlin Heidelberg.

Gradišar, D. and Mušič, G., 2007. Production-process modelling based on production-management data: a Petri-net approach, *International Journal of Computer Integrated Manufacturing*, 20 (8), 794–810.

Han, L., Xing, K., Chen, X., Lei, H. and Wang, F., 2014. Deadlock-free genetic scheduling for flexible manufacturing systems using Petri nets and deadlock controllers, *International Journal of Production Research*, 52 (5), 1557–1572.

Huang, H., Du, H. and Ahmad, F., 2013. Job shop scheduling with Petri nets, *in* P. Pardalos (ed.), *Handbook of Combinatorial Optimization*, Springer, pp. 1667–1711.

Jensen, K., 1997. *Coloured Petri Nets: Basic Concepts,Analysis Methods and Practical Use*, Vol. 1, 2 edn, Springer-Verlag, Berlin.

Kurz, M. E. and Askin, R. G., 2004. Scheduling flexible flow lines with sequence-dependent setup times, *Eur. Journal of Operational Research*, 159 (1), 66–82.

Lakos, C. and Petrucci, L., 2007. Modular state space exploration for timed Petri nets, *International Journal on Software Tools for Technology Transfer*, 9, 393–411.

Löscher, T., Mušič, G. and Breitenecker, F., 2007. Optimisation of scheduling problems based on timed Petri nets, *6th EUROSIM Congress on Modelling and Simulation*, Vol. II, Ljubljana, Slovenia.

Mejía, G., Montoya, C., Cardona, J. and Castro, A. L., 2012. Petri nets and genetic algorithms for complex manufacturing systems scheduling, *International Journal of Production Research*, 50 (3), 791–803.

Mušič, G., 2011. Efficient exploration of coloured Petri net based scheduling problem solutions, *23th European Modeling & Simulation Symposium*, Rome, Italy, pp. 681–689.

Mušič, G., 2012a. Efficient exploration and evaluation of Coloured Petri net based scheduling problem solutions, *14th IFAC Symposium on Information Control Problems in Manufacturing, INCOM'12*, IFAC, Bucharest, pp. 150–155.

Mušič, G., 2012b. Schedule optimization based on Coloured Petri nets and local search, *7th Vienna Conference on Mathematical Modelling, full paper preprint volume*, ARGESIM, Vienna.

Mušič, G., 2015. Generation of feasible Petri net based scheduling problem solutions, *IFAC-PapersOnLine*, 48 (1), 856 – 861. 8th Vienna International Conference on Mathematical Modelling MATHMOD 2015.

Napalkova, L., Piera, M. A. and Mušič, G., 2014. Performance evaluation of flexible manufacturing systems by coloured timed Petri nets and timed state space generation, *in* J. Campos, C. Seatzu and X. Xie (eds), *Formal Methods in Manufacturing*, CRC Press, pp. 381–408.

Piera, M. A. and Mušič, G., 2011. Coloured Petri net scheduling models: Timed state space exploration shortages, *Math.Comput.Simul.*, 82, 428–441.

Pinedo, M. L., 2008. *Scheduling: Theory, Algorithms, and Systems*, 3rd edn, Springer Publishing Company.

Recalde, L., Silva, M., Ezpeleta, J. and Teruel, E., 2004. Petri nets and manufacturing systems: An examples-driven tour, *in* J. Desel, W. Reisig and G. Rozenberg (eds), *Lectures on Concurrency and Petri Nets*, Vol. 3098 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 71–89.

Ríos-Mercado, R. Z. and Bard, J. F., 1998. Computational experience with a branch-and-cut algorithm for flowshop scheduling with setups, *Computers & Operations Research*, 25 (5), 351 – 366.

Sindičić, I., Petrović, T. and Bogdan, S., 2012. Machine-job incidence matrix based analysis of manufacturing systems in time domain, *IEEE International Conference on Control Applications (CCA)*, pp. 234–239.

Tuncel, G. and Bayhan, G. M., 2007. Applications of Petri nets in production scheduling: a review, *International Journal of Advanced Manufacturing Technology*, 34, 762–773.

Viswanadham, N. and Narahari, Y., 1992. *Performance Modeling of Automated Manufacturing Systems*, Prentice-Hall, Englewood Cliffs, USA.

Werner, F., 2013. A survey of genetic algorithms for shop scheduling problems, *in* P. Siarry (ed.), *Heuristics: Theory and Applications*, Nova Science Publishers, pp. 161–222.

Xing, K., Han, L., Zhou, M. and Wang, F., 2012. Deadlock-free genetic scheduling algorithm for automated manufacturing systems based on deadlock control policy, *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 42 (3), 603–615.

AUTHOR BIOGRAPHY

GAŠPER MUŠIČ received B.Sc., M.Sc. and Ph.D. degrees in electrical engineering from the University of Ljubljana, Slovenia in 1992, 1995, and 1998, respectively. He is Full Professor at the Faculty of Electrical Engineering, University of Ljubljana. His research interests are in discrete event and hybrid dynamical systems, supervisory control, planning, scheduling, and industrial informatics. His Web page can be found at `http://msc.fe.uni-lj.si/Staff.asp`.