Timed Petri-Net Representation for Short Term Scheduling of Multiproduct Batch Plants

Tianlong Gu¹, and Parisa A. Bahri

School of Engineering, Murdoch University (Rockingham Campus), Murdoch, WA 6150, Australia
parisa@eng.murdoch.edu.au (To whom all correspondence should be addressed)
¹School of Computer Engineering, Guilin Institute of Electronic Technology, Guilin 541004, P. R. of China
cligu@mango.glet.edu.cn

Abstract: Scheduling is one of the interesting research fields in batch chemical industries, which substantially differs from that in discrete manufacturing systems. One of the main differences between these two fields is the intermediate storage of materials, which appears in chemical processes. This situation prevents the direct applicability of some of the scheduling techniques (such as Petri nets) used for discrete manufacturing systems, in batch chemical plants. In this paper, a timed Petri nets-based formulation for scheduling of multiproduct batch chemical plants with unlimited intermediate storage policy is presented. To find the best schedule, a modified branch-and-bound and TPPNs execution (MBBTE) solution algorithm is proposed, which will be examined through an illustrative example.

Keywords: scheduling, batch plants, branch and bound, timed Petri nets, discrete event systems

1. Introduction

Economics as well as technological advantages have resulted in batch processes being predominantly used for low volume, high-value-added products in many chemical industries such as food, pharmaceuticals, specialty chemicals etc. This is mostly due to the flexibility and capability of producing a great number of different products through using the same set of equipment in these plants. Batch processes can be distinguished from continuous processes by the discrete nature of the events (operations or tasks) that occur in the plant. When products are with high degree of similarity, they may need the same processing steps, and hence pass through the same series of processing units. Plants producing such products are called multiproduct plants. However, when products are with low degree of similarity, they may need different processing steps, and hence visit different series of processing units. Such plants are referred to as multipurpose plants. In other words, a multiproduct plant utilises a single equipment configuration to process all the products in a sequential fashion, while in a multipurpose plant, production is achieved through multiple production lines, each of which is dedicated to the production of one product. In a batch plant, intermediate storage between processing units is also important to maintain smooth flow of materials and to satisfy the requirements of processing recipes. The different types of intermediate storage policies are unlimited intermediate storage (UIS), no intermediate storage (NIS), finite intermediate storage (FIS), zero wait (ZW), and mixed intermediate storage (MIS) [1][2].

Production scheduling is an important aspect of batch process operations to achieve high productivity and operability. In this case, the main issues of study are the development of algorithms for the completion time and the optimal scheduling. The completion time algorithm is to calculate the completion time of each product on each processing unit, [3]. The optimal scheduling is to find the most efficient way to utilise the limited resources within the scheduling time period in order to fulfil production objectives. In most cases, the objective of optimal scheduling is to solve the best sequence of production to minimise the total operational time, the so-called makespan. The optimal scheduling is commonly formulated as a mixed integer linear or nonlinear programming (MILP, MINLP) problem, [4][5]. These methodologies allow the incorporation of almost any constraint in the problem formulation, but the solution algorithms suffer from the combinatorial complexity, especially if nonlinearities are included in the model. To improve the computational efficiency, heuristic search algorithms are often adopted [6], which tend towards finding the optimal solutions. The major problem with these algorithms is that they usually end up in local optima. As a remedy, the stochastic global optimisation techniques (simulated annealing and genetic algorithms) have been proposed [7][8]. These algorithms have proved to be efficient in the optimisation problems that are dominated by a combinatorial element; furthermore, they are not as much affected by nonlinearities and complex objective functions as mathematical programming algorithms. However, it is quite difficult to include complex constraints into the internal representation to ensure the feasibility of the generated schedules. To easily formulate the scheduling problem, a combinatorial technique using the graph representation has been proposed [9][10]. This technique is then extended to the short term scheduling of multipurpose batch plants [11]. However, the main problem with this method is that it can only handle the UIS and NIS storage policies.

Petri nets approach has proved to be a promising technique to solve many difficult problems associated with the modelling, formal analysis, and design of discrete event systems [12]. Petri nets have traditionally been used to model and analyse discrete manufacturing systems [13][14][15], through representing simple production lines with buffers, machine shops, flexible manufacturing systems, automated
assembly lines, intelligent machines and robotics, and implementing supervisory control of their logical behaviours. Several studies on timed Petri-net-based scheduling of manufacturing systems were also reported [15][16][17][18]. More recently, Petri nets techniques have been developed for the logic modelling and coordination control of discrete batch operations [19][20][21]. Although Petri net has shown to be an effective tool for formulating and solving the scheduling problems of discrete manufacturing systems, it has not been given much attention in scheduling of batch processes. Therefore, it is the aim of this paper to present a timed Petri nets-based approach for scheduling of multiproduct batch plants. First, some backgrounds on timed Petri nets are given. Then a representation for scheduling of multiproduct batch plants considering unlimited intermediate storage policy will be introduced. Finally a novel search algorithm, which combines the execution of timed Petri nets with modified branch-and-bound technique to find the optimal makespan will be proposed. The applicability of the proposed approach will be tested by means of several examples.

2. What are Timed Petri Nets
An ordinary Petri-net is a bipartite directed graph represented by a finite set of places, finite set of transitions, input arcs and output arcs. A marked Petri net contains tokens in addition to the elements described. The tokens, places and transitions must be assigned a meaning for proper interpretation of the model. They are traditionally interpreted in the following way:

- **Places** represent resources such as unit operations. The existence of one or more tokens in a place represents the availability of a particular resource.
- A **transition firing** represents an activity. This activity begins and ends with two consecutive events. The time between events may be zero, corresponding to an intermediate transition.
- **Places** and **transitions** together represent conditions and precedence relations in the system's operation. For example, a token in a place (resulting in a marked place) can imply that the condition is true, and no token, that it is false.

The marking of a place $p$ in ordinary Petri nets is a mapping of the place to a nonnegative integer representing the number of tokens sitting in this place, denoted by $m(p)$. The markings of all the places in an ordinary Petri net constitute the marking vector $m$, whose $i$-th component $m(p_i)$ represents the number of tokens in the $i$-th place $p_i$. The initial marking is denoted by $m_0$.

To consider the amount of time that it takes for an operation, time can be incorporated into Petri-nets by attaching time (generally in the form of deterministic time delay) to either transitions, resulting in timed transition Petri net (TTPN) or to the places, resulting in a timed place Petri net (TPPN).

3. Timed Petri-Net-Based Representation for Batch Plant Scheduling
In a multiproduct batch plant, there are $n$ products to be produced in $m$ processing units. For each product, the sequence of visiting the processing units is specified, referred to as the product (or job) routing. The processing time $t_j$ for product (or job) $i$ ($i = 1, 2, ..., n$) in unit $j$ ($j = 1, 2, ..., m$) is usually given. There is a total number of $(m \times n)$ processing activities in the plant.

In this research, TPPNs are used to model the batch operations. Consider $O_j$ to represent a processing activity of the $i$-th job being performed in unit $j$. We can use two transitions $t_{nj}$ and $t_{nj}'$ to represent the start and the termination of this operation, and one place $p_i$ with time duration $t_j$ to represent the operation being performed if there is one token in the place. Now the operations in a multiproduct batch plant can be easily modelled using the following procedures:

**Step 1:** For the $i$-th job, each processing activity is represented by two transitions $t_{nj}$, $t_{nj}'$, and one place $p_i$ (representing the processing activity).
**Step 2:** For the $i$-th job, introduce the $i$-th initial marked place $p_i$, representing the beginning of the job, and the $i$-th final place $p_{if}$, representing the end of the $i$-th job;
**Step 3:** For the processing activity $O_j$, $j \leq m$, introduce the intermediate place $p_{in}$, representing the readiness of the intermediate product for next processing activity, $O_{jn-1}$;
**Step 4:** For the processing unit $j$, introduce resource place $p_j$ representing the unit's availability;
**Step 5:** In terms of job routing, all the activities involved in the $i$-th job are linked, and modelled as a TPPN sub-model;
**Step 6:** Through merging the sharing resource places $p_j$ of all the processing units, the sub-models of all the jobs are interconnected, and the complete TPPN model for the batch plants scheduling is developed.

We illustrate the above procedure by considering a case study with unlimited intermediate storage policy that consists of four products ($p_1$ to $p_4$), all of which need three processing steps carried out by three processing units ($u_1$ to $u_3$). The processing times of each product in each unit are given in Table 1.

<table>
<thead>
<tr>
<th>Units</th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
</tr>
</thead>
<tbody>
<tr>
<td>u1</td>
<td>3.5</td>
<td>4.0</td>
<td>3.5</td>
<td>12.0</td>
</tr>
<tr>
<td>u2</td>
<td>4.3</td>
<td>5.5</td>
<td>7.5</td>
<td>3.5</td>
</tr>
<tr>
<td>u3</td>
<td>8.7</td>
<td>3.5</td>
<td>6.0</td>
<td>8.0</td>
</tr>
</tbody>
</table>

For the $i$-th job, we can construct the TPPN sub-model shown in Figure 1a, in which the time duration of each place is put in the brackets. The complete TPPN model of this batch plant is developed by merging the common resource places $p_j$ (see Figure 1b).
4. Modified Branch-and-Bound and TPPNs Execution (MBBTE) Algorithm

Once all operations in a plant are represented by a TPPN model, the computation of all possible future markings of the place starting from the initial marking can be drawn in the form of reachability tree. The best schedule can then be estimated by generating that portion of the tree that is necessary for optimal search. This will be performed by developing a modified Branch-and-Bound algorithm. The algorithm has two critical steps: (1) extending, and (2) checking.

Extending: For the current node (marking $m_j$), find all the possible enabled transitions and compute their new marking $m_{j+1}$ in terms of enabled conditions and firing rules of TPPNs. These new markings $\{m_{j+i}\}$ will be the descendant or new nodes extended from the current node (marking $m_j$), and will be put in the OPEN list. Here, we use $\{m_{j+i}\}$ to represent the set of all the next possible markings. The important information to be recorded here, are: the time that the new marking is produced, denoted by $e(m_{j+i})$, the duration that each processing unit is utilised, denoted by $PT_j(m_{j+i})$, and the return pointer of $m_j$ as the current node (marking $m_j$).

Checking: For the current node (final marking $m_f$), determine whether the current upper bound of the objective function should be updated or not; If the current node (marking $m_f$) is not a final marking, determine whether a branch should be extended (forward search) or discarded (backward search). This is done based on the current upper bound of objective, which is the estimated value of the schedules associated with this node (branch), denoted by $h(m_j)$. The upper bound (UB) of the objective can be initialised as a large number $M$ so as to ensure that reasonably optimal makespan is not left out. When a feasible schedule is obtained, the upper bound is updated. In the case of objective function being the schedule's makespan, estimated value of the objective at the current node (marking $m_f$) can be calculated by the following equations:

$$g(m_f) = \max_j \left( \sum \tau_j - PT_j(m_f) \right),$$

$$h(m_f) = e(m_f) + g(m_f).$$

where, $\sum \tau_j - PT_j(m_f)$ is the total duration of processing time of those jobs which are assumed to be processed consecutively in unit $j$. For the schedules associated with these branches through current node (marking $m_f$), $h(m_f)$ is the lower bound of the makespan.

The modified branch-and-bound and TPPNs execution (MBBTE) algorithm for short-term scheduling of multiproduct batch plants with make span as the objective can be presented as follows:

**s1**: Initialize the upper bound (UB) of makespan as a large number $M$, and $e(m_f)$ and $PT_j(m_f)$ equal to 0; Store the initial marking $m_i$ in OPEN list. Here, OPEN is a stack, where items are stored and retrieved in a last-in-first-out order;
s2: If OPEN is empty, the search is complete, and the firing sequence associated with times and makespan would be the final result; then Stop;
s3: Retrieve marking \( m_i \) from OPEN, and let \( m_i \) be the current node (marking);
s4: If \( m_i \) is not the final marking, Go to s6;
s5: If \( e(m_i) < UB \), update \( UB = e(m_i) \); Go to s2;
s6: For the current marking \( m_i \), compute \( g(m_i) = \max \{ \sum_i(t_i) - PT(m_i) \} \), and \( h(m_i) = e(m_i) + g(m_i) \);
s7: If \( h(m_i) \geq \) upper bound of makespan, go to s2;
s8: For the current marking \( m_i \), find all the possible enabled transitions, and compute the new marking \( m_{i+} \) for each of them, and store them in the OPEN list;
s9: Set the return pointer of \( m_{i+} \); Update \( e(m_{i+}) \) and \( PT(m_{i+}) \); Go to s2.

5. Case Study

To illustrate the implementation of the MBBTE algorithm, we consider a simple multiproduct batch plant consisted of 2 units, which can process 3 products with UIS policy. The processing times of each product in each unit are given in Table 2.

<table>
<thead>
<tr>
<th>Units</th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
</tr>
</thead>
<tbody>
<tr>
<td>u1</td>
<td>3.0</td>
<td>4.0</td>
<td>3.0</td>
</tr>
<tr>
<td>u2</td>
<td>4.0</td>
<td>5.0</td>
<td>7.0</td>
</tr>
</tbody>
</table>

The corresponding TTPNs model is shown in Figure 2.

![Figure 2: The TTPNs model](image)

In this TTPN model, the initial and final markings were \( m_0 = (1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0) \) and \( m_f = (0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0) \), respectively. The algorithm converged after 21 iterations and 2 min CPU time. The optimal searches of the reachability tree are shown in Figure 3.

![Figure 3: The optimal searches of the reachability tree](image)

As shown in Figure 3, the optimal firing sequence is: \( m_0 \)
\( t_{11}, m_1, t_{11}, m_2, t_{11}, m_3, t_{11}, m_4, t_{11}, m_5, t_{11}, m_6, t_{11}, m_7, t_{11}, m_8, t_{11}, m_9, t_{11}, m_{10}, t_{11}, m_{11}, t_{11}, m_{12}, t_{11}, m_{13}, t_{11}, m_{14}, t_{11}, m_{15}, t_{11}, m_{16}, t_{11}, m_{17}, t_{11}, m_{18}, t_{11}, m_{19} \). Hence, the optimal sequence (Job3, Job2, Job1) can be drawn as a Gantt chart, as in Figure 4.

4095
6. Conclusions

This paper has presented a Timed-Placed Petri-Net (TPPN) model for scheduling of multiproduct batch plants. It has been shown that the marking changes of the TPPN model completely describe the evolution of different operations, in the presence of unlimited intermediate storage policy. Hence, the optimal schedule can be found by searching the reachability tree of the model by the proposed modified branch and bound and TPPN execution (MBBTE) algorithm. The scheduling of a batch plant consisted of two units and three products has been used to show the applicability of the proposed technique.

It is understood from this research that the Petri-net based approach has a great potential for solving a variety of batch scheduling problems. In this regard, further research is under way to cover the following aspects:

- Application of this approach to the batch plants with various intermediate storage policies, such as unlimited intermediate storage (UIS), no intermediate storage (NIS), finite intermediate storage (FIS), and mixed intermediate storage (MIS).
- Application of the proposed methodology to different scheduling problems (such as multi-purpose batch plant scheduling)
- Comparison between this technique and other methodologies.

7. References
