Batch Processing with Sequence Dependent Setup Times: Using Precedence Graph

Petr Vilím

Charles University
Faculty of Mathematics and Physics
Malostranské náměstí 2/25, Praha 1, Czech Republic
vilim@kti.mff.cuni.cz

Abstract. Filtering (constraint propagation) turned out to be an efficient method how to reduce the search space for backtrack-based algorithms. In this paper I propose the use of precedence graph for scheduling of batches with sequence dependent setup times. Domain filtering based on this graph can improve existing filtering algorithms. I describe how to easily detect all precedences found by algorithm edge-finding and how to maintain transitive closure of the precedence graph in reasonable time. All the results of this paper hold also for usual disjunctive scheduling because it is the special case of batch processing with sequence dependent setup times.

1 Introduction

There are four filtering algorithms for batch processing with sequence dependent setup times: edge finding, not-first/not-last, not-before/not-after [5] and sequence composition [4]. Each of these algorithms filters out different inconsistencies therefore they can be used together to get maximum pruning. In this paper I show that it is possible to achieve even better filtering when these algorithms are combined with filtering based on precedence graph.

Batch processing with sequence dependent setup times is an extension of classical disjunctive scheduling (for definition of disjunctive scheduling see for example [1]). We have to schedule the set of activities T on one resource with the following characteristics:

- Each activity $i \in T$ has its' release time r_i and due time d_i. Processing of the activity cannot start before the release time r_i and cannot complete after the due time d_i. The role of a filtering algorithm is to reduce the intervals $\langle r_i, d_i \rangle$ and this way prune the search space.
- Each activity $i \in T$ has a family (type) f_i. The set of all the families is F.
- Only the activities with the same family can be processed together. Activities processed together form a batch – its processing start together and complete together.
- Processing time of an activity $i \in T$ depends only on the family f_i. Let p_{f_i} denotes this processing time, for short also p_i.

– Sum of the capacities c_i of activities in the batch cannot exceed the capacity of the resource C. I.e. the resource is renewable.
– After the batch completes there is a setup time needed before another batch can start. This setup time depends on the families of both consequent batches. When the first batch have family f and the second one family g then s_{fg} denotes minimum setup time between them.
– There is no setup needed between the batches with the same family:

$$\forall f \in F : \; s_{ff} = 0$$

– Setup time also meet the triangle inequality:

$$\forall f, g, h \in F : \; s_{fg} + s_{gh} \geq s_{fh}$$

Often we need to deal with a subset of activities $\Omega \subseteq T$. I quickly establish the notation for Ω from [5]. Processing of Ω can start at first in the time $r_\Omega = \min\{r_i, i \in \Omega\}$ and cannot end after the time $d_\Omega = \max\{d_i, i \in \Omega\}$. Let $u(\Omega)$ be the minimal pure processing time needed for activities Ω. (When computing $u(\Omega)$ I do not care about the setup times, release and due times.) Let F_Ω be the set of all families of Ω. Minimal setup time needed for processing Ω is $s(F_\Omega)$. When the processing of Ω have to start with a family $f \in F_\Omega$ then the minimum setup time needed is $s(f, \Omega)$. Minimum time needed for processing Ω is $p(\Omega) = u(\Omega) + s(F_\Omega)$. Function s can be precomputed in the time $O(k^2 2^k)$. For the details how these functions can be computed see [5].

The only one from the four mentioned algorithms which detects new precedences is edge-finding. Therefore I briefly resume it here.

Consider an arbitrary set $\Omega \subset T$ and an activity $i \in (T \setminus \Omega)$. When the activity i is scheduled before the set Ω the processing of Ω can start at first in the time $r_i + p_i$. The setups for this processing is $s(f_i, F_\Omega \cup \{f_i\})$. If such processing of Ω ends after the time d_Ω then the activity i cannot be scheduled before the set Ω. So we get following *not-before* rule:

$$\forall \Omega \subset T, \; \forall i \in (T \setminus \Omega) : \\ r_i + p_i + s(f_i, F_\Omega \cup \{f_i\}) + u(\Omega) > d_\Omega \quad \Rightarrow \quad i \not\ll \Omega \tag{1}$$

Similarly, next rule says that when the activity i cannot be scheduled between the activities Ω it have to be scheduled before whole set Ω or after it:

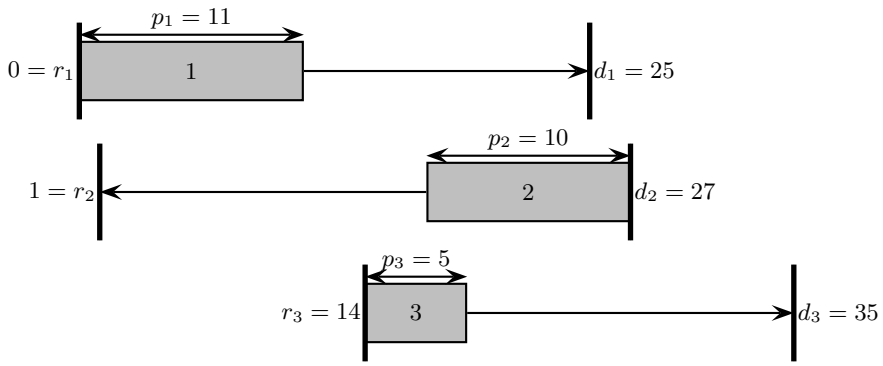
$$\forall \Omega \subset T, \; \forall i \in (T \setminus \Omega) : \\ d_\Omega - r_\Omega < p(\Omega \cup \{i\}) \quad \Rightarrow \quad (i \ll \Omega \text{ or } \Omega \ll i) \tag{2}$$

When both the rules (1), (2) hold then $\Omega \ll i$. Resulting time bond adjustment is:

$$\Omega \ll i \quad \Rightarrow \quad r_i \geq \max\{r_{\Omega'} + u(\Omega') + s(F_{\Omega'} \cup \{f_i\}, f_i), \;\; \Omega' \subseteq \Omega\} \tag{3}$$

There are also symmetric rules which deal with precedences $i \ll \Omega$. Edge finding algorithm enforce all the changes from these rules.

Following picture is an example of an disjunctive scheduling problem (disjunctive scheduling is special case of batch scheduling). None of the discussed algorithms (i.e. edge-finding, not-first/not-last, not-before/not-after and sequence composition) find any time bound adjustment for this problem:



Edge-finding recognize that the activity number 1 have to be processed before the activity number 3, i.e. $1 \ll 3$, and similarly $2 \ll 3$. Still, each of these precedences alone is weak – it do not enforce any change of the time bounds. However from the knowledge $\{1, 2\} \ll 3$ we can deduce $r_3 = r_1 + p_1 + p_2 = 21$. This is the motivation for my effort to maintain all of the precedences in the precedence graph.

2 Precedence Graph

The easiest way how to find new precedences is the disjunctive constraint. It says that if an activity i cannot be processed before nor together with an activity j it have to be processed after the activity j. For batch processing with sequence dependent setup times it means that $i \ll j$ when following conditions do not hold:

$$f_i = f_j \quad \& \quad c_i + c_j \leq C \quad \& \quad \min\{d_i, d_j\} - \max\{r_i, r_j\} \geq p_i \tag{4}$$

$$r_j + p_j + s_{f_j f_i} \leq d_i - p_i \tag{5}$$

The first condition holds when the activities i, j can be processed together and the second one when the activity j can be processed before the activity i. When the both conditions (4) and (5) do not hold then I say that the precedence $i \ll j$ is *detectable*.

Now I show that all the precedences found by the algorithm edge-finding can be subsequently found by the disjunctive constraint:

Proposition 1. *When edge-finding does is not able to find further time bound adjustments then all the precedences which edge-finding found are detectable.*

Proof. Let us suppose that edge-finding proved $\Omega \ll j$. I show that for an arbitrary activity $i \in \Omega$ edge-finding made r_j big enough for the precedence $i \ll j$ being *detectable*.

Edge-finding proved $\Omega \ll j$ so the condition (2) holds:

$$\begin{aligned} p(\Omega \cup \{j\}) &> d_\Omega - r_\Omega \\ u(\Omega \cup \{j\}) + s(F_\Omega \cup \{f_j\}) &> d_\Omega - r_\Omega \\ r_\Omega &> d_\Omega - u(\Omega \cup \{j\}) - s(F_\Omega \cup \{f_j\}) \end{aligned} \tag{6}$$

Because edge-finding is not able to further change the time bounds according to (3):

$$\begin{aligned} r_j &\geq \max\{r_{\Omega'} + u(\Omega') + s(F_{\Omega'} \cup \{f_j\}, f_j),\ \Omega' \subseteq \Omega\} \\ r_j &\geq r_\Omega + u(\Omega) + s(F_\Omega \cup \{f_j\}, f_j) \end{aligned}$$

r_Ω can be replaced by the right side of the inequality (6):

$$r_j > d_\Omega - u(\Omega \cup \{j\}) - s(F_\Omega \cup \{f_j\}) + u(\Omega) + s(F_\Omega \cup \{f_j\}, f_j)$$

And because:

$$\begin{aligned} u(\Omega) - u(\Omega \cup \{j\}) &\geq -p_i \\ s(F_\Omega \cup \{f_j\}, f_j) - s(F_\Omega \cup \{f_j\}) &\geq 0 \\ d_\Omega &\geq d_i \end{aligned}$$

following inequality holds:

$$r_j > d_i - p_j$$

And thus neither condition (4) nor (5) hold and the precedence $i \ll j$ is *detectable*. □

The filtering based on precedence graph is simple: build up the set Ω of all activities which have to be processed before the task i and then use the rule (3):

Sort the activities according to r_i
for $i \in T$ **do begin**
  $\Omega := \emptyset$;
  $m := -\infty$;
  for $j \in T$ in decreasing order or r_j **do**
    if $j \ll i$ **then begin**
      $\Omega := \Omega \cup \{j\}$;
      $m := \max(m,\ r_j + u(\Omega) + s(F_\Omega \cup \{f_i\}, f_i))$;
    end;
  r_i:$-\max(m, r_i)$;
end;

Note that there is also a symmetric version of this algorithm for precedences $i \ll \Omega$.

3 Transitive Closure

Several authors suggest to compute the transitive closure of the precedence graph (e.g. [3], [2]). However computing transitive closure is time-consuming (e.g. Floyd-Warshall algorithm is $O(n^3)$). After the addition of a new precedence into the graph the correction of the transitive closure can be made in the time $O(n^2)$. Still the number of newly discovered precedences can be theoretically even $O(n^2)$.

In this section I show that the correction of the transitive closure have to be made only after the addition of *nondetectable* precedence.

When the precedence $i \ll j$ is *propagated* by edge-finding or precedence graph, new time bounds fulfill following inequalities (special cases of the rule (3) and its' symmetric version for $\Omega = \{i\}$ and $\Omega = \{j\}$):

$$r_j \geq r_i + p_i + s_{f_i f_j}$$
$$d_i \leq d_j - p_j - s_{f_i f_j}$$

Proposition 2. *Let $a \ll b$, $b \ll c$ and one of those precedences is detectable and the second one is propagated. Then the precedence $a \ll c$ is detectable.*

Proof. We distinguish two cases:

1. **$a \ll b$ is detectable and $b \ll c$ is propagated.**
   because $b \ll c$ is *propagated*:

   $$r_c - s_{f_b f_c} \geq r_b + p_b$$

   and because $a \ll b$ is *detectable*:

   $$r_b + p_b + s_{f_b f_a} > d_a - p_a$$
   $$r_c - s_{f_b f_c} + s_{f_b f_a} > d_a - p_a$$

   Together with triangle inequality for setup times $s_{f_b f_c} + s_{f_c f_a} \geq s_{f_b f_a}$:

   $$r_c + s_{f_c f_a} > d_a - p_a$$

   Thus both the conditions (4), (5) cannot hold and the precedence $a \ll c$ is *detectable*.
2. **$a \ll b$ is propagated and $b \ll c$ is detectable.**
   Because $a \ll b$ is *propagated*:

   $$d_b - p_b \geq d_a + s_{f_a f_b}$$

   And because the second precedence $b \ll c$ is *detectable*:

   $$r_c + p_c + s_{f_c f_b} > d_b - p_b$$
   $$r_c + p_c + s_{f_c f_b} > d_a + s_{f_a f_b}$$

We use the triangle inequality again, this time $s_{f_c f_a} + s_{f_a f_b} \geq s_{f_c f_b}$:

$$r_c + p_c > d_a - s_{f_c f_a}$$

Once again the conditions (4) and (5) do not hold and the precedence $i \ll j$ is *detectable*. □

According to this proposition: when precedence propagating do not yield any further adjustments then it made the transitive closure of all *detectable* precedences with the rest of them. The only missing precedences are in the transitive closure of *nondetectable* precedences. And when we add only $O(1)$ such precedences in each search step (e.g. as search decision), we can repair this transitive closure in the time $O(n^2)$.

4 Conclusions

According to my experimental results[1], filtering based on precedence graph can improve the filtering for batch processing with sequence dependent setup times, even when there are no *nondetectable* precedences. Cheap maintaining of full transitive closure of the precedence graph can be also useful for other filtering algorithms or some search heuristics. Note that all the results of this paper can be also used for classical disjunctive scheduling, because it is the special case of batch processing with sequence dependent setup times ($C = c_i = 1$, $s_{fg} = 0$).

References

[1] Philippe Baptiste and Claude Le Pape. Edge-finding constraint propagation algorithms for disjunctive and cumulative scheduling. In *Proceedings of the Fifteenth Workshop of the U.K. Planning Special Interest Group*, 1996.

[2] Peter Brucker. Complex scheduling problems, 1999. URL citeseer.nj.nec.com/brucker99complex.html.

[3] W. Nuijten F. Foccaci, P. Laborie. Solving scheduling problems with setup times and alternative resources. In *Proceedings of the 4th International Conference on AI Planning and Scheduling, AIPS'00*, pages 92–101, 2000.

[4] P. Vilím and R. Barták. A filtering algorithm sequence composition for batch processing with sequence dependent setup times. Technical Report 2002/1, Charles University, Faculty of Mathematics and Physics, 2002.

[5] P. Vilím and R. Barták. Filtering algorithms for batch processing with sequence dependent setup times. In *Proceedings of the 6th International Conference on AI Planning and Scheduling, AIPS'02*, 2002.

[1] Not included here because of space limitation