Online Supplement to "Validation Sequence Optimization: A Theoretical Approach"

Gediminas Adomavicius
Department of Information and Decision Sciences, Carlson School of Management, University of Minnesota, 321 19th Avenue South, Minneapolis, Minnesota 55455, USA, gedas@umn.edu

Alexander Tuzhilin
Department of Information, Operations, and Management Sciences, Stern School of Business, New York University, 44 West 4th Street, New York, New York 10012, USA, atuzhili@stern.nyu.edu

INFORMS Journal on Computing

Appendix A: NP-Hardness Result for Validation Sequence Optimization

Task Sequencing to Minimize Weighted Completion Time The following problem is often referred to as the problem of "Task Sequencing on a Single Processor to Minimize Weighted Completion Time" (Garey and Johnson 1979).

Assume that a set of tasks T has to be sequenced for processing by a single machine. The sequencing of the tasks must obey the precedence constraints imposed by a given directed acyclic graph $G = (V, E)$, where each vertex $v \in V$ is associated with a different task (therefore, $|T| = |V|$). In other words, G imposes a partial order on T. Task $t' \in T$ must precede task $t'' \in T$ if there is a directed path from t' to t'' in G.

Furthermore, each task t is assigned a processing time $p(t) \in Z^+$ and a weight $w(t) \in Z$. Given a specific sequencing of T, e.g., $s =< t_1, \ldots, t_k >$, the completion time of each task t_i is denoted as $C(t_i)$ and can be calculated as

$$C(t_i) = \sum_{j=1}^{i} p(t_j) \tag{19}$$

where we assume that the processing of the first task begins immediately (i.e., at time 0) and there is no idle time between consecutive jobs.

The objective of the sequencing problem is to find the feasible sequence (i.e., that obeys the partial order imposed by G) $s =< t_1, \ldots, t_k >$ ($t_i \in T$) that minimizes the weighted total completion time $WTCT(s)$, defined as a weighted sum of individual completion times, i.e.,

$$WTCT(s) = \sum_{i=1}^{k} w(t_i)\ C(t_i) \tag{20}$$

Lawler (1978) showed that the above problem is NP-hard. Furthermore, it was also shown that the above problem remains NP-hard even when all $w(t) = 1$.

Assuming $w(t) = 1$ for all $t \in T$ and using the definition of $C(t)$ from (19), the weighted total completion time of sequence $s =< t_1, \ldots, t_k >$ can be expressed as

$$WTCT(s) = \sum_{i=1}^{k} C(t_i) = \sum_{i=1}^{k} \sum_{j=1}^{i} p(t_j) = \sum_{i=1}^{k} (k + 1 - i)\ p(t_i) \tag{21}$$

Equivalence of the Two Problems As indicated above, the problem of finding the task sequence that obeys the specified partial order and *minimizes* the weighted total completion time is NP-hard. We will show that the problem of finding the task sequence that obeys the specified partial order and *maximizes* the weighted total completion time is also NP-hard.

Let $G = (V, E)$ be an acyclic directed graph representing the partial order to be imposed on tasks T. Then we will define a "reverse" graph $G' = (V', E')$ as follows. Let $V' = V$ and let E' contain the same edges as E, only each edge should point in the reverse direction. That is, $E' := \{(u, v) : (v, u) \in E\}$.

As indicated in the following lemma, it can be shown that $s =< t_1, \ldots, t_k >$ minimizes $WTCT$ with respect to partial order G if and only if $s' =< t_k, \ldots, t_1 >$ (i.e., s' is the reversed sequence s) maximizes $WTCT$ with respect to partial order G'.

Lemma 22 *$s =< t_1, \ldots, t_k >$ minimizes $WTCT$ with respect to partial order $G \iff s' =< t_k, \ldots, t_1 >$ maximizes $WTCT$ with respect to partial order G'.*

The above lemma indicates that solving the problem of task sequencing to minimize weighted completion time subject to partial order constraints is equivalent to solving the problem of task sequencing to maximize weighted completion time subject to partial order constraints. Since the former problem has been shown to be NP-hard (Lawler 1978), the latter problem is NP-hard as well. In addition, the latter problem is equivalent to our restricted validation sequence optimization (i.e., benefit maximization) problem (12), since in both cases we are searching for the sequence that satisfies the given partial order and maximizes essentially the same function. (The functions in the two problems differ only by a constant that does not depend on a particular sequencing and, therefore, does not affect the solution.) Hence, our restricted optimization problem is NP-hard as well.

Appendix B: Proofs of Main Theoretical Results

Proof of Theorem 4

▶ First, let's assume that validation sequences s and s' contain validation operators u and v that satisfy all three conditions. We will show that $s \not\sim s'$.

Based on condition 3, there exists an input element e that satisfies Boolean expression (7). Because this expression is a conjunction of several subexpressions, e satisfies each of these subexpressions. Based on this we derive the following.

Since both $p_u(e)$ and $\bigwedge_{i=1}^{x-1} \neg p_i(e)$ hold, we have that e satisfies predicate p_u (which is at position x in s), but does not satisfy any of the predicates $p_1, p_2, \ldots, p_{x-1}$. These predicates are at positions $1, 2, \ldots, x-1$ respectively in sequence s, therefore all predicates that precede p_u in the validation sequence would not match e. Consequently, in the sequence s, e would be matched by predicate p_u and labeled with l_u. Obviously, $u \prec_s v$, since otherwise v (and not u, as we just showed) would match the input e in sequence s.

Similarly, since both $p_v(e)$ and $\bigwedge_{j=1}^{y-1} \neg p'_j(e)$ hold, we have that e satisfies predicate p_v (which is at position y in s'), but does not satisfy any of the predicates $p'_1, p'_2, \ldots, p'_{y-1}$. Therefore, in the sequence s', e would be matched by operator p_v and labeled with l_v. Obviously, $v \prec_{s'} v$, since otherwise u (and not v, as we just showed) would match the input e in sequence s'.

Since both $u \prec_s v$ and $v \prec_{s'} u$, condition 1 is satisfied automatically.

Based on the condition 2, $l_u \neq l_v$. Therefore, s would validate e differently than s'. Therefore, when $D = \{e\}$, we have $VI_s(D) \neq VI_{s'}(D)$. Hence, $s \not\sim s'$.

Conversely, let's assume that $s \not\sim s'$. We will show that these sequences contain validation operators u and v that satisfy all three conditions mentioned above.

$s \not\sim s' \Rightarrow (\exists D)(VI_s(D) \neq VI_{s'}(D))$. Let's denote $VI_s(D) = (V_1, V_2, \ldots, V_{|\mathcal{L}|})$ and $VI_{s'}(D) = (V'_1, V'_2, \ldots, V'_{|\mathcal{L}|})$. Here V_i ($i = 1, \ldots, |\mathcal{L}|$) is a subset of D labeled with the label L_i by sequence s. Similarly, V'_i ($i = 1, \ldots, |\mathcal{L}|$) is a subset of D labeled with the label L_i by sequence s'. Since $VI_s(D) \neq VI_{s'}(D)$, we have that $(V_1, \ldots, V_{|\mathcal{L}|}) \neq (V'_1, \ldots, V'_{|\mathcal{L}|})$. Therefore, there exists i such that $V_i \neq V'_i$.

Since $V_i \neq V'_i$, let's assume (without loss of generality) that there exists an entity $e \in D$ such that $e \in V_i$, but $e \notin V'_i$. (It could also be $e \in V'_i$ and $e \notin V_i$, in which case the proof would be virtually the same as below.) Since $e \notin V'_i$, there exists $j \in \{1, \ldots, |\mathcal{L}|\}$ such that $i \neq j$ and $e \in V'_j$. Note that e can not remain unvalidated by s', as shown in Lemma 3.

Because $e \in V_i$, there must exist a validation operator $u = (L_i, p_u)$ in the sequence s (say, at the position x, i.e., $pos_s(u) = x$) that validates e (i.e., $p_u(e)$ is `True`), but none of the preceding operators do (i.e., $\neg p_i(e)$ for all $i = 1, \ldots, x-1$). Therefore, both $p_u(e)$ and $\bigwedge_{i=1}^{x-1} \neg p_i(e)$ hold.

Similarly, because $e \in V'_j$, there must exist a validation operator $v = (L_j, p_v)$ in the sequence s' (say, at the position y, i.e., $pos_s(v) = y$) that validates e (i.e., $p_v(e)$ is `True`), but none of the preceding operators do (i.e., $\neg p'_j(e)$ for all $j = 1, \ldots, y-1$). Therefore, both $p_v(e)$ and $\bigwedge_{i=1}^{y-1} \neg p'_j(e)$ hold.

The previous two paragraphs combined show that condition 3 holds. Condition 2 also holds, since u and v operators described above have different labels (i.e., L_i and L_j, $i \neq j$). Finally, condition 1 holds as well, because the same input element e is validated by u in sequence s and by operator v in sequence s', which would be impossible when either of operators u and v precedes the other one in both sequences, since they both match e. ◀

Proof of Theorem 8

▶ Let x be the *largest* number from the set $\{1, \ldots, k\}$, such that $o_{x+1} \prec_{s'} o_x$. Then, let's construct the sequence $s'' =< o''_1, \ldots, o''_k >$ as follows. Let $o''_i := o_i$, for all $i = \{1, \ldots, k\}$, such that $i \neq x$ and $i \neq x+1$. Also, let $o''_x = o_{x+1}$ and $o''_{x+1} = o_x$.

Essentially, sequence s'' is the same as s except for o_x and o_{x+1} that are swapped. Obviously, s'' is a simple permutation of s, thus $dist(s, s'') = 1$.

Now we will show that $s \sim s''$. Since s'' is a simple permutation of s, Corollary 6 gives us two conditions to be satisfied in order to have $s \sim s''$.

Assume o_x and o_{x+1} have the same label, i.e., $l_x = l_{x+1}$, then the first condition from Corollary 6 is satisfied. Therefore, $s \sim s''$. In case o_x and o_{x+1} do not have the same label, the only way for $s \sim s''$ to be true is for o_x and o_{x+1} to satisfy the second condition from Corollary 6. For the remainder of this proof we will assume that o_x and o_{x+1} do not have the same label, and we will show that they satisfy the second condition from Corollary 6, i.e.,

$$\neg(p_x \wedge p_{x+1}) \vee \bigvee_{i=1}^{x-1} p_i \tag{22}$$

Let's go back to sequences s and s' for a moment. Since $s \sim s'$, from Corollary 5 we have that *all* pairs of operators from s, including o_x and o_{x+1}, must satisfy at least one of the three necessary and sufficient conditions for $s \sim s'$. Let's consider the pair o_x and o_{x+1}.

Since o_x precedes o_{x+1} in s, but o_{x+1} precedes o_x in s' (that's how we chose o_x in the beginning of the proof), the first condition is not satisfied by these two operators. These operators do not satisfy the second condition as well, since they do not have the same label (according to our assumption). Therefore, since $s \sim s'$, o_x and o_{x+1} satisfy the third condition of Corollary 5, namely:

$$\neg(p_x \wedge p_{x+1}) \vee \bigvee_{i=1}^{x-1} p_i \vee \bigvee_{j=1}^{y-1} p'_j \tag{23}$$

where y is the position of o_{x+1} in s'. Therefore, $o_{x+1} = o'_y$. Also note that, by construction, none of o'_j ($j \in \{1, \ldots, y-1\}$) can be equal to o_x or o_{x+1}, since $o_{x+1} = o'_y$ and $o_{x+1} \prec_{s'} o_x$.

We will show that every o'_j ($j \in \{1, \ldots, y-1\}$) is from among $o_1, \ldots, o_{x-1}$. Suppose otherwise, there exists $j \in \{1, \ldots, y-1\}$ such that $o'_j = o_z$, where $x \leq z \leq k$. Since, as mentioned above, none of o'_j ($j \in \{1, \ldots, y-1\}$) can be equal to o_x or o_{x+1}, we can obtain an even tighter bound for z, i.e., $x+1 < z \leq k$.

Then, consider validation operators o_{x+1} and o_z. o_{x+1} precedes o_z in s, because $x+1 < z$. However, o_z precedes o_{x+1} in s', because $pos_{s'}(o_{x+1}) = y$ and $pos_{s'}(o_z) < y$. From Lemma 7 we have, that there exists t, $x+1 \leq t \leq z-1$, such that o_{t+1} precedes o_t in s'.

Thus, we showed that there exists $t \geq x+1 > x$, such that o_{t+1} precedes o_t in s'. However, by definition x is the *largest* number, such that o_{x+1} precedes o_x in s' (i.e., we chose x to be the largest such number in the first paragraph of this proof). We derived a contradiction, therefore our assumption that there exists $j \in \{1, \ldots, y-1\}$ such that $o'_j = o_z$, where $x \leq z \leq k$ is incorrect. This implies that every o'_j ($j \in \{1, \ldots, y-1\}$) is from among $o_1, \ldots, o_{x-1}$. Therefore, every p'_j ($j \in \{1, \ldots, y-1\}$) is from among $p_1, \ldots, p_{x-1}$. Consequently, the third necessary condition (23) of $s \sim s'$ in this case is equivalent to:

$$\neg(p_x \wedge p_{x+1}) \vee \bigvee_{i=1}^{x-1} p_i \tag{24}$$

Hence, based on the fact that $s \sim s'$, we proved that the Boolean expression (24) holds for every element $e \in \mathcal{E}$. However, this expression is exactly the same as the one described by (22), which was needed to prove that $s \sim s''$ (when o_x and o_{x+1} do not have the same label). Therefore, o_x and o_{x+1} satisfy the second sufficient condition of Corollary 6 and, hence, $s \sim s''$.

Now we have $s \sim s'$ and $s \sim s''$. Because of the transitivity and the symmetry of the relation $R_\sim$ (see Lemma 2), $s' \sim s''$ is also true. Also, we know that $dist(s, s') = d$ and $dist(s, s'') = 1$. Because of how we constructed s'', s' has all the same precedence inversions with respect to s'' as with respect to s, except for one. More specifically, o_x and o_{x+1} have the same precedence in both s' and s''. Therefore, the distance between s' and s'' is one less than between s' and s, i.e., $dist(s', s'') = d - 1$. ◀

Proof of Theorem 9

▶ Assume $s \sim s'$. Let's denote $s_0 := s$ and $s_d := s'$. Based on Theorem 8, there exists sequence s_1, such that s_1 is a safe simple permutation of s_0, and also $s_1 \sim s_d$, and $dist(s_1, s_d) = d - 1$. Repeat this process for s_1 and s_d to obtain s_2, etc. In general, when we have s_i, such that $s_i \sim s_d$ and $dist(s_i, s_d) = d - i$, we can obtain s_{i+1} (which is a safe simple permutation of s_i), such that $s_{i+1} \sim s_d$ and $dist(s_{i+1}, s_d) = d - i - 1$. Hence, there exists $d + 1$ validation sequences s_0, s_1, …, s_d, such that $s_0 = s$, $s_d = s'$, and s_i is a safe simple permutation of s_{i-1} for every $i = 1, \ldots, d$.

Conversely, assume that there exists $d + 1$ validation sequences s_0, s_1, …, s_d, such that s_i is a safe simple permutation of s_{i-1} (i.e., $s_{i-1} \sim s_i$) for every $i = 1, \ldots, d$. By transitivity of the equivalence relation: $s_0 \sim s_d$. Hence, $s \sim s'$. ◀

Proof of Lemma 19

▶ Let's assume $s \approxeq s'$ and let's consider an arbitrary validation operator o_i from sequence s, i.e., $pos_s(o_i) = i$. Also, let $j = pos_{s'}(o_i)$. We have to show that $n_i = n'_j$. We will show this by showing that o_i validates exactly the same subset of D in both s and s'.

Assume otherwise, that there exists $e \in D$ such that either (a) o_i validates e in s but not in s', or (b) o_i validates e in s' but not in s. We will provide the proof for the first of these two situations. The proof for the second one is essentially identical.

Since there exists $e \in D$ such that o_i validates e in s but not in s', there must exist a validation operator o_x such that $o_x \prec_{s'} o_i$ and o_x validates e. However, $o_i \prec_s o_x$, because otherwise o_i would not be able to validate e in s (i.e., o_x would validate e before o_i). Therefore, we have two validation operators o_i and o_x such that $o_i \prec_s o_x$, $o_x \prec_{s'} o_i$, and $p_i \not\perp p_x$ (since there exists $e \in D$ that can be validated by both o_i and o_x). This is a contradiction, because by the definition of very strong equivalence all pairs of validation operators must satisfy one of two conditions (see Definition 10 in the paper), whereas the pair o_i and o_x satisfies neither.

Therefore, given s and s', where s' is very strongly equivalent to s, each validation operator validates exactly the same subset of inputs from D in both s and s'. Hence, for all i: $n_i = n'_j$, where $j = pos_{s'}(o_i)$. ◀

Proof of Theorem 21

▶ First, it is clear that $n_i = n'_i$ for all $i < x$, since only the operators o_x and o_{x+1} are permuted. That is, first $x - 1$ operators in both sequences s and s' are the same and will produce the same validation results.

It is also easy to see that $n_i = n'_i$ for all $i > x+1$. This is the case because the set of first $x+1$ validation operators is the same in both sequences (not necessarily in the same order). Obviously, the exact same subset of input dataset D would remain unvalidated after $x+1$ operators in both. (For more precise reasoning, consider the two sequences of length $x+1$ and see Lemma 3 in the paper.) In addition, $o_i = o'_i$ for $i > x+1$. Therefore, $n_i = n'_i$ for all $i > x+1$.

We still need to estimate n'_x and n'_{x+1}. Consider operators $o_x = (l_x, p_x)$ and $o_{x+1} = (l_{x+1}, p_{x+1})$. We know that $o'_x = o_{x+1}$ and $o'_{x+1} = o_x$. Since $s \approx s'$ and s' is a simple permutation of s, according to Lemma 16 we have one of the following two possibilities:

- $p_x \perp p_{x+1}$. This means that validation operators o_x and o_{x+1} can never both match the same input data point. Therefore, it does not matter whether o_x precedes o_{x+1} (as in sequence s) or o_{x+1} precedes o_x (as in sequence s'), they will still validate the same exact data points as before. Hence, $n'_x = n_{x+1}$ and $n'_{x+1} = n_x$.

- $l_x = l_{x+1}$. Since o_{x+1} will precede o_x in sequence s', obviously, it will be able to validate at least as many data points in s' as in s, therefore $n'_x \geq n_{x+1}$. As mentioned above,the set of first $x+1$ validation operators is the same in both sequences (not necessarily in the same order) and the exact same subset of input dataset D would remain unvalidated after $x+1$ operators in both. Therefore, $\sum_{i=1}^{x+1} n_i = \sum_{i=1}^{x+1} n'_i$. However, since $n_i = n'_i$ for $(i < x)$, we have that $n_x + n_{x+1} = n'_x + n'_{x+1}$. Furthermore, since $n'_x \geq n_{x+1}$ (as we have just shown), we have that $n'_{x+1} \leq n_x$.

Therefore, in both cases above it is true that $n_x + n_{x+1} = n'_x + n'_{x+1}$ and $n'_{x+1} \leq n_x$. Now, let's estimate how much different is the cost of sequence s from the cost of sequence s', when $s \approx s'$ and s' is a simple permutation of s.

In the case where s' is a simple permutation of s, we get (by applying the above analysis and also by plugging in the definition of the *benefit* function from Equation 4):

$$\begin{aligned}
\Delta_{s\to s'} &= \mathit{benefit}(s', D) - \mathit{benefit}(s, D) \\
&= \sum_{i=1}^{k-1} (k-i)\ n'_i\ -\ \sum_{i=1}^{k-1} (k-i)\ n_i = \sum_{i=1}^{k-1} (k-i)(n'_i - n_i) \\
&= (k-x)(n'_x - n_x) + (k-x-1)(n'_{x+1} - n_{x+1}) \\
&= (k-x)(n'_x + n'_{x+1} - n_x - n_{x+1}) + (n_{x+1} - n'_{x+1}) \\
&= n_{x+1} - n'_{x+1} \\
&\geq n_{x+1} - n_x
\end{aligned}$$

where in the last equation we actually have an equality in the case when p_x and p_{x+1} are orthogonal, as demonstrated in Lemma 19. ◀

References

Garey, M. R., D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, NY.

Lawler, E. L. 1978. Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Ann. of Discrete Math.* **2** 75–90.