Available online at www.sciencedirect.com

ScienceDirect

Computers & Industrial Engineering 53 (2007) 394–400

computers & industrial engineering

www.elsevier.com/locate/dsw

Batch scheduling to minimize the weighted number of tardy jobs ☆

Erdal Erel [a,*], Jay B. Ghosh [b]

[a] *Faculty of Business Administration, Bilkent University, 06800 Bilkent, Ankara, Turkey*
[b] *Apratech, LLC, Los Angeles, CA, USA*

Received 1 August 2005; accepted 15 March 2007
Available online 13 May 2007

Abstract

In this paper, we address a single-machine scheduling problem with due dates and batch setup times to minimize the weighted number of tardy jobs. We give a pseudo-polynomial dynamic program and a fully-polynomial approximation scheme for the case where the due dates are uniform within a family.

Keywords: Scheduling; Batch setup times; Dynamic programming; Approximation

1. Introduction

Suppose that we have a single machine that can process at most one job at a time. Suppose also that a set of N independent jobs is ready at time zero that requires uninterrupted processing on the machine. The jobs belong to F different families, family f having N_f jobs (so that $\sum_{1 \leqslant f \leqslant F} N_f = N$). Job j of family f has a processing time p_{fj}, a due date d_{fj} and a weight w_{fj}. A setup time s_f in incurred whenever the machine changes over to the processing of a family f job from a job that belongs to another family. Let C_{fj} be the completion time of job j of family f in some schedule. The job is called *tardy* if $C_{fj} > d_{fj}$; this is indicated by setting $U_{fj} = 1$. Similarly, the job is called *early* if $C_{fj} \leqslant d_{fj}$; this is indicated by $U_{fj} = 0$. (Note that we follow the so-called *item-availability* model in that we assume that all jobs are available as soon as they are completed, without having to wait until the completion of other jobs in their families.) Our objective is to schedule all the jobs on the machine so that $\sum_{1 \leqslant f \leqslant F} \sum_{1 \leqslant j \leqslant N_f} w_{fj} U_{fj}$ is minimized. In other words, we seek to minimize the *weighted number of tardy jobs*. Extending the standard notation, this is the $1/s_f/\sum w_{fj} U_{fj}$ problem. The interested reader is referred to Webster and Baker (1995) for a comprehensive overview of batch scheduling problems of this and other kinds.

☆ This manuscript was processed by Area Editor Subhash C. Sarin.

* Corresponding author. Tel.: +90 312 266 4164; fax: +90 312 266 4958.
E-mail address: erel@bilkent.edu.tr (E. Erel).

0360-8352/$ - see front matter
doi:10.1016/j.cie.2007.03.006

It follows from Bruno and Downey (1978) that the $1/s_f/\sum w_{fj}U_{fj}$ problem is NP-hard at least in the ordinary sense. Monma and Potts (1989) presented a dynamic programming solution for this problem that is pseudo-polynomial only for a fixed F. A special case of the $1/s_f/\sum w_{fj}U_{fj}$ problem is realized if we assume that $d_{fj} = d_f$ for all j and f (that is, all jobs within a family have identical due dates). We call this as $1/s_f, d_{fj} = d_f/\sum w_{fj}U_{fj}$ problem. It follows from Karp (1972) that this problem is also NP-hard at least in the ordinary sense (as is its extreme special case of $1/s_f, d_{fj} = d, p_{fj} = p, w_{fj} = w_f/\sum w_{fj}U_{fj}$ problem). Rote and Woeginger (1998) have recently addressed the easier $1/s_f, d_{fj} = d_f/\sum U_{fj}$ problem and shown that it is $O(N^2)$ solvable. Finally, in the separate context of scheduling on a batch processing machine (where the jobs do not necessarily belong to families but are batched together for processing), Hochbaum and Landy (1994) have studied a similar problem to minimize the weighted number of tardy jobs under the *batch-availability* model. In this model, all the jobs in a batch are processed consecutively and a job is deemed available only when the last job in its batch is completed. This problem is shown to be NP-hard, but it admits a pseudo-polynomial time solution.

In this paper, we address the $1/s_f, d_{fj} = d_f/\sum w_{fj}U_{fj}$ problem in its general form. Specifically we give a $O(N\min\{\sum_f s_f + \sum_f\sum_j p_{fj}, \max_f\{d_f\}, \sum_f\sum_j w_{fj}\})$ pseudo-polynomial dynamic programming solution. We also give a $O(N^2\max\{\log N, 1/\varepsilon\})$ fully-polynomial approximation scheme which delivers a solution within $(1+\varepsilon)$ times the optimal solution for any $\varepsilon > 0$. We note at this point that the $1//\sum w_jU_j$ problem, without family setup times, admits a pseudo-polynomial solution (Lawler & Moore, 1969) and a fully-polynomial approximation scheme (Gens & Levner, 1981) that has complexity orders comparable to those that we have obtained for the $1/s_f, d_{fj} = d_f/\sum w_{fj}U_{fj}$ problem; this shows that the $1/s_f, d_{fj} = d_f/\sum w_{fj}U_{fj}$ problem is not much harder than the $1//\sum w_jU_j$ problem.

2. Dynamic programming solution

Assume, without loss of generality, that the job families are indexed such that $d_1 \leqslant d_2 \leqslant \ldots \leqslant d_F$ and that the jobs within family f, $1 \leqslant f \leqslant F$, are indexed such that $p_{f1}/w_{f1} \leqslant p_{f2}/w_{f2} \leqslant \ldots \leqslant p_{fN_f}/w_{fN_f}$. Assume also that all the parameters are integer valued. For future use, let $S = \sum_f s_f$, $P = \sum_f\sum_j p_{fj}$, $D = \max_f\{d_f\}$ and $W = \sum_f\sum_j w_{fj}$. We have already let $N = \sum_{1 \leqslant f \leqslant F} N_f$.

It is known (Monma & Potts, 1989) that the main challenge here is the identification of the *early* jobs that are scheduled without any inserted idle time in the *earliest due date* (EDD) order; the scheduling of the *tardy* jobs is inconsequential. Two observations are important to the development of our dynamic program (DP) and the fully-polynomial approximation scheme. The first observation, which is easily proved through a simple job exchange argument, has also been made for the $1/s_f, d_{fj} = d_f/\sum U_{fj}$ problem (Rote & Woeginger, 1998). This observation holds as well for the $1/s_f, d_{fj} = d_f/\max\{w_{fj}U_{fj}\}$ problem (we will call this problem as MinMax hereafter), which we use for developing lower and upper bounds on the optimal solution of the $1/s_f, d_{fj} = d_f/\sum w_{fj}U_{fj}$ problem (this problem will be called as MinSum) and whose DP solution we describe below in parallel.

Observation 1: (General) There is an optimal schedule where the early jobs have at most one setup per family and are processed without any inserted idle time in the EDD order.

We describe the DP in an enumerative form. In our enumeration, we build a schedule starting with job 1 of family 1 (stage 1) and ending with job N_F of family F (stage N). At stage k, $1 \leqslant k \leqslant N$, we consider if we should schedule job j of family f as an *early* or a *tardy* job (clearly, $\sum_{1 \leqslant g \leqslant f-1} N_g < k \leqslant \sum_{1 \leqslant g \leqslant f} N_g$). At the end of stage k, let $\mathbf{L}_k$ be the set of early jobs and $\mathbf{R}_k$ be the set of tardy jobs; note that $\mathbf{L}_k$ and $\mathbf{R}_k$ are mutually exclusive and that $|\mathbf{L}_k| + |\mathbf{R}_k| = k$. Also, let T_k be the total processing and setup times of the jobs in $\mathbf{L}_k$, n_k be the number of jobs from family f that are in $\mathbf{L}_k$. Finally, let $W^{\mathbf{R}}{}_k$ be the total weight of the jobs in $\mathbf{R}_k$ (and $w^{\mathbf{R}}{}_k$ the maximum weight among these jobs). To solve MinSum, we minimize $W^{\mathbf{R}}{}_N$ over all complete schedules (and $w^{\mathbf{R}}{}_k$ for MinMax). For MinSum, the triple $\langle T_k, n_k, W^{\mathbf{R}}{}_k\rangle$ adequately represents a partial schedule at the end of stage k (as does $\langle T_k, n_k, w^{\mathbf{R}}{}_k\rangle$ for MinMax). Letting $\delta(x) = 1$ if $x > 0$ and 0 otherwise, we can now state our second observation (which helps us identify a minimal set of nondominated partial schedules at each stage of DP). The observation is easily proved through a straightforward identical completion argument.

Observation 2: (For MinSum): Given two partial schedules, represented respectively by the triples $\langle T_k, n_k, W^{\mathbf{R}}{}_k\rangle$ and $\langle T'_k, n'_k, W'^{\mathbf{R}}{}_k\rangle$ at stage k of DP, such that $T_k \leqslant T'_k$, $\delta(n_k) = \delta(n'_k)$ and $W^{\mathbf{R}}{}_k \leqslant W'^{\mathbf{R}}{}_k$, it is sufficient to retain only $\langle T_k, n_k, W^{\mathbf{R}}{}_k\rangle$ for further enumeration.

(For MinMax): Given the triples $\langle T_k, n_k, w^{\mathbf{R}}{}_k\rangle$ and $\langle T'_k, n'_k, w'^{\mathbf{R}}{}_k\rangle$ at stage k of DP, such that $T_k \leqslant T'_k, \delta(n_k) = \delta(n'_k)$ and $w^{\mathbf{R}}{}_k \leqslant w'^{\mathbf{R}}{}_k$, it is sufficient to retain only $\langle T_k, n_k, w^{\mathbf{R}}{}_k\rangle$.

Letting Ω_k be the set of partial schedules (equivalently, triples) at stage k of DP and UB be a known upper bound on the minimum of $W^{\mathbf{R}}{}_N$ (MinSum) or $w^{\mathbf{R}}{}_N$ (MinMax) over all complete schedules, we can state the following procedure.

Procedure DP_MinSum (DP_MinMax):

Step 0: Set $\Omega_0 = \{\langle 0,0,0\rangle\}$ and $k = 0$.

Step 1: For $f = 1, \ldots, F$ and $j = 1, \ldots, N_f$:

(a) Set $k = k + 1$.

(b) Reset the 2nd co-ordinate (n_{k-1}) of all triples in Ω_{k-1} to 0 if $j = 1$.

(c) For each triple in Ω_{k-1}, create 2 new triples and add temporarily to Ω_k:

(For MinSum)

(i) $\langle T_{k-1}, n_{k-1}, W^{\mathbf{R}}{}_{k-1} + w_{fj}\rangle$.

(ii) $\langle T_{k-1} + s_f + p_{fj}, n_{k-1} + 1, W^{\mathbf{R}}{}_{k-1}\rangle$ if $\delta(n_{k-1}) = 0$ and $\langle T_{k-1} + p_{fj}, n_{k-1} + 1, W^{\mathbf{R}}{}_{k-1}\rangle$ otherwise.

(For MinMax)

(i) $\langle T_{k-1}, n_{k-1}, \max\{w^{\mathbf{R}}{}_{k-1}, w_{fj}\}\rangle$.

(ii) $\langle T_{k-1} + s_f + p_{fj}, n_{k-1} + 1, w^{\mathbf{R}}{}_{k-1}\rangle$ if $\delta(n_{k-1}) = 0$ and $\langle T_{k-1} + p_f j, n_{k-1} + 1, w^{\mathbf{R}}{}_{k-1}\rangle$ otherwise.

(d) Delete, from Ω_k, all triples with $T_k > d_f$.

(e) Delete, from Ω_k, all triples with $W^{\mathbf{R}}{}_k$ (or $w^{\mathbf{R}}{}_k$) > UB.

(f) Extract a minimal set of nondominated triples from among the survivors in the current Ω_k and let this set be the final Ω_k (cf., Observation 2).

Step 2: From Ω_N, find the triple with the minimum $W^{\mathbf{R}}{}_N$ (MinSum) or $w^{\mathbf{R}}{}_N$ (MinMax).

The above procedure is exact as it never discards a partial schedule that upon completion may lead to an optimal schedule (unless there is another equivalent or better partial schedule) and it counts the weighted number of tardy jobs correctly.

As for the complexity of MinSum, note that for each value of $\delta(n_k)$, the procedure retains at most one triple for each distinct value of T_k or $W^{\mathbf{R}}{}_k$ at stage k. The number of values of T_k, $W^{\mathbf{R}}{}_k$, $\delta(n_k)$ are bounded by $\min\{S + P, D\}$, UB, and 2, respectively. Note that UB $\leqslant W$. The cardinality of Ω_k is thus $O(\min\{S + P, D, W\})$. Over N stages, this translates to $O(N\min\{S + P, D, W\})$.

Similarly, for MinMax, the number of triples retained at stage k depends on the distinct values of T_k or $w^{\mathbf{R}}{}_k$ and $\delta(n_k)$. Note that $w^{\mathbf{R}}{}_k$ takes on at most N distinct values. The cardinality of Ω_k in this case is $O(\min\{S + P, N\})$ or $O(N)$ and the overall complexity is $O(N^2)$.

In sum, Procedure DP_MinSum solves the $1/s_f, d_{fj} = d_f/\sum w_{fj}U_{fj}$ problem exactly in $O(N\min\{S + P, D, W\})$ time. Also, Procedure DP_MinMax solves the $1/s_f, d_{fj} = d_f/\max\{w_{fj}U_{fj}\}$ problem in $O(N^2)$ time. An example to illustrate the procedure is given in Appendix A.

2.1. Remarks on the DP solution

1. If $S + P$, D or W is bounded above by a polynomial function poly(N) of N, we immediately have a $O(N\text{poly}(N))$ algorithm for MinSum. Thus, for the $1/s_f, d_{fj} = d_f/\sum U_{fj}$ problem, where $W = N$, we get a $O(N^2)$ solution as in Rote and Woeginger (1998).
2. There are other cases of MinSum where a poly(n) bound on W may apply such as when the w_{fj} is a notional penalty. For example, when there is a designated subset of jobs that should not be tardy, it suffices to assign these jobs a weight of N and the rest a weight of 1. This case is solved in $O(N^3)$ time.
3. Finally, the inclusion of n_k in the description of a partial schedule makes DP quite versatile. It can now handle side constraints such as those on the maximum number of jobs per family that can be tardy.

3. Fully-polynomial approximation scheme

Our fully-polynomial approximation scheme (FPAS) for MinSum is obtained through a simple modification of the DP_MinSum procedure. Let OPT be the unknown optimal solution value of MinSum and let LB be a lower bound on OPT. Note that UB is an upper bound on OPT as before. Let $\mathrm{WMAX}^{\mathbf{R}}{}_k = \max\{W^{\mathbf{R}}{}_k\}$, where the maximum is taken over all triples in Ω_k at the end of Step 1(e) in the DP_MinSum procedure. Finally, let $\Delta = (\mathrm{eLB})/(N)$. The basic procedure (which delivers a solution value within eLB of OPT) can now be described as follows.

Procedure APPROX_MinSum:

All Steps (0–2) are identical to Procedure DP_MinSum except for Step 1(f). Replace Step 1(f) with the following:

Step 1′(f):

(i) Divide the interval $[0, \mathrm{WMAX}^{\mathbf{R}}{}_k]$ into subintervals of width Δ.

(ii) From all the triples in Ω_k that have $W^{\mathbf{R}}{}_k$ in a given subinterval and the same $\delta(n_k)$, retain in the final Ω_k one with the smallest T_k.

At stage k of Procedure APPROX_MinSum, the number of subintervals is $\lceil \mathrm{WMAX}^{\mathbf{R}}{}_k/\Delta \rceil$. Substituting for Δ, we have:

$$\lceil \mathrm{WMAX}^{\mathbf{R}}{}_k/\Delta \rceil \leqslant (N/e)(\mathrm{WMAX}^{\mathbf{R}}{}_k/\mathrm{LB}) + 1 \leqslant (N/e)(\mathrm{UB}/\mathrm{LB}) + 1.$$

Note that $\mathrm{WMAX}^{\mathbf{R}}{}_k \leqslant \mathrm{UB}$ follows from Step 1(e) of Procedure DP_MinSum. Each subinterval retains at most two triples and thus the cardinality of Ω_k at the end of Step 1′(f), after the modification, is $O((N/e)(\mathrm{UB}/\mathrm{LB}))$. Over N stages, we can then say that Procedure APPROX_MinSum has time complexity $O((N^2/e)(\mathrm{UB}/\mathrm{LB}))$.

As for the approximation error, keeping Observation 2 in mind, it is clear that the maximum error that is admitted at stage k is bounded by Δ. Over N stages, the maximum total error is thus bounded by ΔN. Substituting for Δ, it is easy to see that the error is bounded by eLB.

3.1. Remarks on the FPAS

1. If UB/LB is bounded above by a constant c or a polynomial poly(N), we immediately get an FPAS for MinSum by setting $e = \varepsilon$; the time complexities are $O(N^2/\varepsilon)$ and $O(N^2\,\mathrm{poly}(N)/\varepsilon)$, respectively. Following Gens and Levner (1981), it is indeed possible to derive such an FPAS (as we will see shortly).
2. Even if a valid lower bound LB on OPT is not available, it is still possible to derive an FPAS for MinSum with $O(N^2 \max\{\log(\varepsilon W), 1/\varepsilon\})$ complexity. This is done by searching for a valid lower bound over the interval $[1/\varepsilon, W]$. We omit the details.
3. Sahni (1976) gives an FPAS for maximizing the *weighted number of early jobs* when there are no batch setup times. For this objective, we can easily get a $O(N^2/\varepsilon)$ FPAS in our case by modifying DP and Procedure APPROX_MinSum. We omit the details.

Let opt be the optimal solution value of MinMax, obtained in $O(N^2)$ time by running Procedure DP_MinMax. The following observation is easily made.

Observation 3: opt $\leqslant$ OPT $\leqslant N$ opt.

Using LB = opt, UB = N opt and $e = \varepsilon$ in Procedure APPROX_MinSum, we immediately get a $O(N^3/\varepsilon)$ FPAS for MinSum. A better complexity can be realized if we are able to tighten the bounds such that UB/LB $\leqslant c$. The following procedure delivers LB and UB with UB/LB $\leqslant 2$ in $O(N^2 \log N)$ time. (In other words, it gives a 2-approximation for MinSum.) Note that, in what follows, LB is always a valid lower bound, while UB is only an estimated upper bound (and remains so until the procedure terminates). Also note that LB > 0, since LB $= 0$ implies that opt $= 0$, which in turn implies that there is an optimal solution to MinMax with a maximum tardiness value of 0 and further that this solution must also be optimal for MinSum as the weighted

number of tardy jobs in this case takes on the minimum possible value of 0 (thus obviating the need for an approximate solution).

Procedure BOUNDS_MinSum:

Step 0: Set LB = opt, UB = 4 opt, $e = 1$ and DONE = '*false*'.

Step 1: Do While (DONE = '*false*'):

(a) Invoke Procedure APPROX_MinSum.

(b) If Procedure APPROX_MinSum delivers a solution, call it APX.

(i) If APX $\leqslant$ 2 LB, set LB = LB and UB = APX.
Else, set LB = APX − LB and UB = APX.

(ii) Set DONE = '*true*'.

Else, set LB = 2 LB and UB = min {N opt, 4 LB}.

Step 2: Deliver LB and UB as the desired bounds.

Notice that UB/LB $\leqslant$ 4 and $e = 1$ in Procedure APPROX_MinSum; therefore, it runs in $O(N^2)$ time. We invoke it at most $\log(N)$ times, since $\log(N) \geqslant \max\{i: 2^i \text{ opt} \leqslant N \text{ opt}\}$. Thus, the overall time complexity of Procedure BOUNDS_MinSum is $O(N^2 \log N)$.

To see that Procedure BOUNDS_MinSum delivers the desired bounds, first note that LB remains a valid lower bound throughout the procedure and that UB upon delivery is valid and satisfies UB/LB $\leqslant 2$. Finally, note that Procedure APPROX_MinSum will always deliver a solution before LB is updated to LB $> N$ opt.

Finally, we state the $O(N^2 \max\{\log N, 1/\varepsilon\})$ FPAS for the $1/s_f, d_{fj} = d_f/\sum w_{fj} U_{fj}$ problem as follows:

Procedure FPAS_MinSum:

Step 1: Call DP_MinMax to get opt; call BOUNDS_ MinSum to get LB and UB.

Step 2: Invoke APPROX_MinSum with e ($= \varepsilon$), LB and UB and deliver the solution.

The procedure is obviously correct. Step 1 takes $O(N^2)$ plus $O(N^2 \log N)$ time and Step 2 takes $O(N^2/\varepsilon)$ time. The complexity of Procedure FPAS_MinSum is thus $O(N^2 \max\{\log N, 1/\varepsilon\})$. The above procedures are also applied to the example problem in Appendix A.

4. Concluding remarks

In this paper, we have addressed the $1/s_f, d_{fj} = d_f/\sum w_{fj} U_{fj}$ single machine scheduling problem, which tries to minimize the weighted number of tardy jobs in presence of family setup times and a uniform due date for each family. The problem can be seen to be NP-hard. We have provided both a $O(N \min\{S + P, D, W\})$ pseudo-polynomial dynamic programming solution and a $O(N^2 \max\{\log N, 1/\varepsilon\})$ fully-polynomial approximation scheme. These results specialize easily to the previously studied cases where there is no setup time and where there is no weight.

Appendix A. An illustrative example

Consider a problem with 2 families, each of which includes 3 jobs. All parameters of the problem are given in Table 1 below.

In Table 2 below, we show how Procedure DP_MinSum works. We assume that we have already obtained a heuristic upper bound UB = 8 (by including jobs 1 and 2 of family 1 and job 1 of family 2 in the early set).

Table 1
Problem parameters

Family	Job						Due date	Setup time
	1		2		3			
	Time	Weight	Time	Weight	Time	Weight		
1	1	1	3	2	6	3	19	10
2	4	3	5	2	8	3	30	12

Table 2
Stage-by-stage description of procedure DP_MinSum

Set of all generated triples						
Stage 0	Stage 1 ($f=1, j=1$)	Stage 2 ($f=1, j=2$)	Stage 3 ($f=1, j=3$)	Stage 4 ($f=2, j=1$)	Stage 5 ($f=2, j=2$)	Stage 6 ($f=2, j=3$)
$\langle 0,0,0\rangle$	$\langle 0,0,1\rangle$	$\langle 0,0,3\rangle$	$\langle 0,0,6\rangle$	$\langle 0,0,9\rangle$[c]	$\langle 16,1,8\rangle$	$\langle 16,1,11\rangle$[c]
	$\langle 11,1,0\rangle$[a]	$\langle 13,1,1\rangle$[a]	$\langle 16,1,3\rangle$[a,d]	$\langle 16,1,6\rangle$[a]	$\langle 21,2,6\rangle$[a]	$\langle 24,2,8\rangle$[a]
		$\langle 11,1,2\rangle$	$\langle 13,1,4\rangle$	$\langle 13,0,7\rangle$	$\langle 13,0,9\rangle$[c]	$\langle 21,2,9\rangle$[c]
		$\langle 14,2,0\rangle$[a]	$\langle 19,2,1\rangle$[a]	$\langle 29,1,4\rangle$[a]	$\langle 30,1,7\rangle$[a,d]	$\langle 29,3,6\rangle$[a]
			$\langle 11,1,5\rangle$	$\langle 19,0,4\rangle$	$\langle 29,1,6\rangle$[d]	$\langle 19,0,9\rangle$[c]
			$\langle 17,2,2\rangle$[a]	$\langle 35,1,1\rangle$[a,b]	$\langle 34,2,4\rangle$[a,b]	$\langle 39,1,6\rangle$[a,b]
			$\langle 14,2,3\rangle$	$\langle 11,0,8\rangle$	$\langle 19,0,6\rangle$	$\langle 11,0,13\rangle$[c]
			$\langle 20,3,0\rangle$[a,b]	$\langle 27,1,5\rangle$[a]	$\langle 36,1,4\rangle$[a,b]	$\langle 31,1,10\rangle$[a,b]
				$\langle 17,0,5\rangle$	$\langle 11,0,10\rangle$	$\langle 27,1,10\rangle$[c]
				$\langle 33,1,2\rangle$[a,b]	$\langle 28,1,8\rangle$[a,d]	$\langle 35,2,7\rangle$[a,b]
				$\langle 14,0,6\rangle$	$\langle 27,1,7\rangle$	$\langle 17,0,10\rangle$[c]
				$\langle 30,1,3\rangle$[a]	$\langle 32,2,5\rangle$[a,b]	$\langle 37,1,7\rangle$[a,b]
					$\langle 17,0,7\rangle$	$\langle 14,0,11\rangle$[c]
					$\langle 34,1,5\rangle$[a,b]	$\langle 34,1,8\rangle$[a,b]
					$\langle 14,0,8\rangle$	$\langle 30,1,8\rangle$
					$\langle 31,1,6\rangle$[a,b]	$\langle 38,2,5\rangle$[a,b]
					$\langle 30,1,5\rangle$	
					$\langle 35,2,3\rangle$[a,b]	

[a] The job associated with a given stage is scheduled early.
[b] The triple is eliminated in Step 1(d) of the procedure.
[c] The triple is eliminated in Step 1(e) of the procedure.
[d] The triple is eliminated in Step 1(f) of the procedure.

Three triples, $\langle 24,2,8\rangle$, $\langle 29,3,6\rangle$ and $\langle 30,1,8\rangle$ survive at the end of stage 6. The triple with the minimum W^R_6, $\langle 29,3,6\rangle$, determines the optimal schedule where jobs 1 through 3 of family 2 are scheduled first followed by the family 1 jobs. The family 2 jobs are all early, while the family 1 jobs are all tardy. The weighted number of tardy jobs is 6.

As indicated earlier, Procedure DP_MinMax works in a similar way. The difference is how the 3rd coordinate of the triple is computed. We simply observe that the optimal maximum tardiness in this case is 3 (given, for example, by the heuristic schedule mentioned above), but omit the procedural details.

Coming to Procedure FPAS_MinSum as applied to this problem, we note that BOUNDS_MinSum is invoked first with $e=1$ and $LB=3$. For these values, $\Delta=0.5$. Thus, because of the integrality of the triple coordinates, APPROX_MinSum essentially generates the same set of triples as in Table 2, delivering $\langle 29,3,6\rangle$ as a 2-approximate solution (which, we know from above, is actually optimal) and yielding $LB=3$ and $UB=6$, and comes to termination in the first pass itself. At this point, when APPROX_MinSum is invoked with any $\varepsilon<1$ (leading to $\Delta<0.5$) $\langle 29,3,6\rangle$ is delivered once again as a $(1+\varepsilon)$-approximate solution.

References

Bruno, J., & Downey, P. (1978). Complexity of task sequencing with deadlines, set-up times, and changeover costs. *SIAM Journal of Computing, 7*, 393–404.

Gens, G. V., & Levner, E. V. (1981). Fast approximation algorithms for job sequencing with deadlines. *Discrete Applied Mathematics, 3*, 313–318.

Hochbaum, D. S., & Landy, D. (1994). Scheduling with batching: Minimizing the weighted number of tardy jobs. *Operations Research Letters, 16*, 79–86.

Karp, R. M. (1972). Reducibility among combinatorial problems. In R. E. Miller & J. W. Thatcher (Eds.), *Complexity of Computer Computations* (pp. 85–104). New York: Plenum Press.

Lawler, E. L., & Moore, J. M. (1969). A functional equation and its application to resource allocation and sequencing problems. *Management Science, 16*, 77–84.

Monma, C. L., & Potts, C. N. (1989). On the complexity of scheduling with batch set-up times. *Operations Research, 37*, 798–804.

Rote, G., & Woeginger, G. J. (1998). Minimizing the number of tardy on a single machine with batch setup time. *Acta Cybernetica, 13*, 423–430.

Sahni, S. (1976). Algorithms for scheduling independent tasks. *Journal of the ACM, 23*, 116–127.

Webster, S., & Baker, K. R. (1995). Scheduling groups of jobs on a single machine. *Operations Research, 43*, 692–703.