A column generation algorithm for the single machine parallel batch scheduling problem

Onur Ozturk[1]

Telfer School of Management, University of Ottawa, 55 Laurier Avenue East, Ottawa, ON, Canada K1N 6N5
oozturk@uottawa.ca

Keywords: Batch scheduling, dynamic programming, column generation, truncated solutions

1 Introduction

Batch scheduling is the type of scheduling where jobs are grouped into batches and processed together. In parallel batch scheduling (p-batch), jobs of the same batch are all processed at the same time and in the same machine. In our problem, jobs have different release dates, r_j, different processing times, p_j, and sizes, v_j. Batches have a fixed capacity B that sum of job sizes in a batch must not exceed. Once their composition is determined, batches may also have different processing times such that each batch must be processed at least as long as the longest processing time of jobs included in that batch. We aim to minimize the total flow time, i.e., $\sum C_j$. Inspired from Graham's notation ((Graham, Lawler, Lenstra & Rinnooy Kan 1979)), our problem can be represented as $1/p-Batch, r_j, p_j, v_j, B/\sum C_j$. Most solution methods for p-batch problems in the literature are metaheuristic methods ((Jia, Zhang, Long, Leung, Li & Li 2018)). In this study, we develop a time indexed column generation model capable of finding high quality solutions within short computational times. Inspired from a 0-1 set partitioning model, each column represents a set of jobs to be processed in the same batch at a given instant. At the end of each iteration, we heuristically obtain primal solutions derived by iteratively rounding the linear programming solution of the column generation model.

2 Set partitioning formulation for the Master Problem

The input of the problem is twofold: n jobs, indexed from 1 to n to be batched complying with batch capacity and a total length T before which all batches must be processed at an instant t such that $1 \leq t \leq T$. Thus for each batch b with processing duration p_b and processing starting time t, time indices $t' \in \{t, t+1, ..., t+p_b-1\}$ indicate the consecutive discrete time instants during which batch b is processed. Let $Set_b \in \{b_1, b_2, ..., b_P\}$ be the set of all feasible job assignments and C_b the cost of processing batch b, i.e., the sum of job completion times of batch b. Let a_{jb} be equal to 1 if job j is assigned to batch b ($\forall j \in \{1, ..., n\}$), 0 otherwise. Also let a_{tb} be equal to 1 if batch b is being processed at instant t ($\forall t \in \{n+1, ..., T\}$), 0 otherwise. Then we have the following conditions for a feasible job assignment and batch processing: $\sum_{j=1}^{n} a_{jb} v_j \leq B$, $p_b = max\{a_{jb} * p_k\}$ ($\forall j \leq n$) and $\sum_{t=n+1}^{n+T} a_{tb} = p_b$. Let C_b be the cost of processing batch b, i.e., total flow time of jobs included in batch b. C_b is calculated as $C_b = \sum_{j=1}^{n} a_{jb} * C_j$ where $C_j = max\{t * a_{tb}\}$).

The decision to take is then if a batch b with a precise processing starting time should be included in the solution. Hence, let the binary decision variable x_b be equal to 1 if batch (column) b is part of the solution and 0 otherwise ($b = 1, ..., P$). We can model the master problem as follows:

$$\text{Min} \sum_{b=1}^{P} x_b * C_b \tag{1}$$

$$\text{s.t.} \tag{2}$$

$$\sum_{b=1}^{P} a_{jb} * x_b = 1 \quad j = 1, .., n \tag{3}$$

$$\sum_{b=1}^{P} a_{tb} * x_b \leq 1 \quad t = n+1, .., n+T \tag{4}$$

$$x_b \in \{0, 1\} \quad \forall b = 1, ..., P \tag{5}$$

In the MP model above, the objective function is the minimization of total flow time. The first constraint set assigns each job to a single batch. The second constraint set indicates that a time instant can be occupied by the processing of at most one batch. We relax the integrity condition for variable x_b and solve the linear relaxation of the MP model by column generation. The master problem is initiated with a restricted set of columns where initial batches are generated with a heuristic regardless of job release dates and processing times. This heuristic generates intervals of different lengths covering iteratively 1, 2,..., n jobs. Then jobs whose release dates fall into the same interval are batched together with the first fit decreasing heuristic. Finally, batches are scheduled as if there is a single machine to have a sufficiently large value for T.

3 Column generation for solving the sub-problem

The sub-problem aims to search for the column with the most negative reduced cost and not currently included in the master problem. The sub-problem is an optimization problem whose constraints are defined by the three fundamental dimensions of the original problem: batch sizes cannot be greater than B, a batch cannot be processed before the greatest release date of jobs included in that batch, batch processing time is given by the longest processing time of jobs in the batch. The dual values of the master problem and the cost of processing the new generated batch, i.e., new column, help to determine the objective function of the sub-problem. The objective function is expressed as $\sum_{j=1}^{n} C_j - (\sum_{j=1}^{n} y_j * \pi_j - \sum_{j=n+1}^{n+T} y_j * \pi_j)$ where C_j is the completion time of job j in the column, y_j is the binary variable indicating if job j makes part of the column ($\forall j \in \{1, ..., n\}$) and if instant t is occupied by the processing of the column ($\forall j \in \{n+1, ..., T\}$), finally π_j is the dual variable corresponding to constraint j in the master model ($\forall j \in \{1, ..., T\}$). We can present the sub-problem in the form of the following minimization problem:

$$\text{Min} \sum_{j=1}^{n} C_j - (\sum_{j=1}^{n} y_j * \pi_j - \sum_{j=n+1}^{n+T} y_j * \pi_j) \tag{6}$$

$$\text{s.t.} \tag{7}$$

$$\sum_{j=1}^{n} y_j * v_j \leq B \tag{8}$$

$$r_b \geq y_j * r_j \quad j = 1, .., n \tag{9}$$

$$p_b \geq y_j * p_j \quad j = 1, .., n \tag{10}$$

$$r_b \leq t + Q(1 - y_j) \quad j = n+1, .., n+T,\ t = j - n - 1 \tag{11}$$

$$r_b + p_b - 1 \geq y_j * t \quad j = n+1, .., n+T,\ t = j - n - 1 \tag{12}$$

$$\sum_{j=n+1}^{n+T} y_j = p_b \tag{13}$$

$$C_j \geq r_b + p_b - Q(1 - y_j) \quad j = 1, .., T \tag{14}$$

$$y_j \in \{0, 1\}, C_j \geq 0, r_b \geq 0, p_b \geq 0 \tag{15}$$

Constraint 8 is the capacity constraint. Constraint sets 9 and 10 determine the starting time, r_b and the processing duration, p_b, of the new generate batch b. Constraint sets 11, 12 and 13 ensure the continuity of the processing for p_b units of time. Finally, constraint set 14 sets the flow time value for jobs in the batch.

Solving the sub-problem with the above model is time consuming. Thus, we developed a pseudo-polynomial dynamic programming (DP) algorithm which solves the sub-problem faster than the above integer model for the case of integer job sizes. The idea of the DP algorithm is to decide what should be the length of the batch (i.e., processing duration) and at which instant the batch should be processed. Then, jobs inducing the smallest reduced cost can be found complying with the batch capacity. Hence, the DP algorithm is controlled by four parameters: processing beginning instant t, processing duration p_b, jobs in batch j, capacity use noted cap. Let $f(.)$ be a four dimensional array to enumerate recursively the minimum reduced cost. We define $f(t, p_b, j, cap)$ as the reduced cost of including job j in a batch whose used capacity is cap with processing duration p_b and processing starting instant t. Initially we set $f(t, p_b, j, cap)$ to infinity $\forall t \in \{0, ..., T\}, p_b \in \{p_1, ..., p_n\}, cap \in \{0, ..., B\}, j \in \{1, ..., n\}$. Then, for each different processing time p_b and potential batch processing instant t, we set $f(t, p_b, 0, v_{arg(p_j=p_b)}) = C_{arg(p_j=p_b)} - \pi_{arg(p_j=p_b)} + \sum_{j=n+t}^{n+t+p_b-1} \pi_j$ $\forall r_{arg(p_j=p_b)} \leq t$. The interpretation of this calculation is the following: batch processing time p_b is computed for all different job processing times as long as release date of the job determining p_b is available before (or at) batch processing instant t. The forth dimension of function $f(.)$ is then set to the size of the job determining p_b. Then, for all other jobs which are not currently in the batch, we recursively compute the minimal reduced cost value: $f(t, p_b, j, cap) = min_{\forall j' < j, cap' \leq cap} f(t, p_b, j', cap') + t + p_b - \pi_j$

The smallest reduced cost is then equal to $min_{\forall t, p_b, cap} f(t, p_b, n, cap)$. There are n possible different processing times for p_b and the parameter t is bounded by T. For each different value of j, two for loops can be run for each of j' and cap' which are bounded by n and B, respectively. Thus, the DP can be implemented in $O(n^3 BT)$ time.

4 Obtaining primal solutions

We implement a rounding technique similar to the one applied by (Mourgaya & Vanderbeck 2007). Primal solutions are obtained by truncating the relaxed solutions of the MP model. After solving the linear relaxation of the initial MP by column generation, we lookup columns/batches whose decision variable value is equal one, i.e., $x_b = 1$. If there is such a column, then it is registered as a validated batch. If there is no such column, then we select the column having the largest x_b value. (If there are multiple columns with the same x_b value, tie is broken by selecting the column with the smallest C_b value where C_b is the sum of job completion times in batch b.) Then, that column is selected as a validated batch. After validating a batch, jobs of that batch are erased from the original problem instance and a residual problem is obtained. Afterwards, the same solution methodology is applied to the residual problem until all jobs are batched.

Once all batches are obtained, it is now time to schedule them on the single machine. The problem is equivalent to a single machine problem in the presence of jobs with different processing times, release dates and weights since batches can contain different numbers of jobs. Inspired from Graham's notation, it can be noted as $1/r_j/\sum w_j C_j$ which is also an NP-hard problem ((Belouadah, Posner & Potts 1992)). However, this problem can be efficiently solved with a time indexed modeling as suggested by (Unlu & Mason 2010).

5 Numerical experimentation

We tested instances containing 10 to 50 jobs in the presence arbitrary release dates generated in the following three ways: $r_j \in U[0,5]$, $r_j \in U[0,25]$ and $r_j \in U[0,5*n]$. Job processing times were generated with $p_j \in U[1,10]$ distribution and batch capacity B was set to 5, 25 and 50 while integer job sizes were generated with a U[1, B/2] distribution. For each combination of number of jobs, release date type and batch capacity, 5 problem instances were generated and tested. We used CPLEX 12.8 for all numerical tests.

Table 1. Average optimality gaps

	Cap = 5					Cap = 25					Cap = 50				
Nbr jobs:	10	20	30	40	50	10	20	30	40	50	10	20	30	40	50
$r_j \in U[0,5]$	4.5	3.5	5.4	1.6	3.8	0.9	5	10.2	13	22.5	5.6	8.3	10.4	1.9	4.5
$r_j \in U[0,25]$	0.6	4.5	6.3	5.3	9.1	3.5	7.1	14.2	13.4	21	2.7	4.7	6.9	11.1	19.5
$r_j \in U[0,5*n]$	0	0.9	0.4	0.2	0	2.1	0.1	0.1	0.2	0.2	0	0.5	0.5	0	0

We compared the solution quality of our method to the solutions given by a straightforward mixed integer linear programming model of the problem where the solution time limit is set to 1800 seconds. Table 1 shows the average optimality gaps of our solution method. We see that the hardest instances are those with a large batch capacity in the presence of medium release dates, i.e., $r_j \in U[0,25]$. Other than that the results are promising and most of the time the optimality gap is close to zero when $r_j \in U[0,5n]$. Moreover, the solution times with the column generation model are very small. The largest instances, containing 50 jobs with a batch capacity of 25, are solved in the worst case within 400 seconds of CPU time.

6 Conclusion

We presented in this study a column generation solution method for the parallel batch scheduling of jobs with different release dates, processing times and sizes. Numerical tests show that the proposed solution method is able to give high quality solutions within short computational times. For future work, we aim to accelerate the sub-problem to decrease the overall solution time of the algorithm.

References

Belouadah, H., Posner, M. E. & Potts, C. N. (1992), 'Scheduling with release dates on a single machine to minimize total weighted completion time', *Discrete applied mathematics* **36**(3), 213–231.

Graham, R., Lawler, E., Lenstra, J. & Rinnooy Kan, A. (1979), 'Optimization and approximation in deterministic sequencing and scheduling: a survey', *Annals of Discrete Mathematics* **5**, 287–326.

Jia, Z.-h., Zhang, H., Long, W.-t., Leung, J. Y., Li, K. & Li, W. (2018), 'A meta-heuristic for minimizing total weighted flow time on parallel batch machines', *Computers & Industrial Engineering* **125**, 298–308.

Mourgaya, M. & Vanderbeck, F. (2007), 'Column generation based heuristic for tactical planning in multi-period vehicle routing', *European Journal of Operational Research* **183**(3), 1028–1041.

Unlu, Y. & Mason, S. J. (2010), 'Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems', *Computers & Industrial Engineering* **58**(4), 785–800.