Lecture Notes in Management Science (2016) Vol. 8, 106–112 ISSN 2008-0050 (Print), ISSN 1927-0097 (Online)

Single machine scheduling with two agents for total completion time objectives☆

Yuvraj Gajpal [1] and Hongwei Li [1]
[1] Asper School of Business, University of Manitoba, Winnipeg, Manitoba R3T 5V4 Canada
{yuvraj.gajpal, hongwei.li}@umanitoba.ca

Proc. ICAOR 2016
Rotterdam, The Netherlands

Keywords:
Competing agents
Heuristics
Shortest processing time
Single-machine scheduling
Total completion time
Two-agents

Abstract

This paper proposes heuristics for the single machine scheduling problem, in which two agents are considered. A set of jobs is involved in each agent and processed by only one processing resource. The objective is to minimize the total completion time of jobs in the first agent subject to an upper bound on the total completion time of the jobs for second agent. We propose two heuristics motivated by the shortest processing time (SPT) first rule to solve this problem. For evaluating the performance of proposed heuristics, numerical experiment is performed on randomly generated problem instances. We use the optimal algorithm from existing literature to evaluate the performance of proposed heuristics for small problems up to 20 jobs. We also used bigger problem instances up to 150 jobs to evaluate the performance of heuristics.

Introduction

Scheduling is a decision-making tool, which try to utilize limited resources via optimizing job schedules (Pinedo 2012, Cardoen *et al*. 2009). This paper considers two agents (agent A and B) having a set of jobs to get processed by a single machine. There are two research directions in terms of objective function considered in two agents scheduling problem. In the first direction each agent objective is assigned some weights and the weighted objectives are minimized (Baker *et al*. 2003). The second direction considers the minimizing objective function of one agent subject to an upper bound on the objective function for other agent (Agnetis, 2004; Agnetis, 2009). This paper considers the two agents problem in which the completion time of jobs of one agent is minimized subject to the pre-specified level of upper bound on the total completion time of jobs of other agent. This scheduling problem has been proved by Agnetis *et al*. (2004) as a binary NP-hard.

Literature Review

Two comprehensive and systemic studies on the single-machine with two agents scheduling problems has been published by Baker *et al*. (2003) and Agnetis *et al*. (2004). In multi-agent related scheduling problem, the crucial factor is the "determination of non-dominated schedule" (Agnetis, Pacciarelli *et al*. 2007, Wan, Vakati *et al*. 2010). Agnetis *et al*. (2004) illustrated polynomial time algorithm and NP-hardness proof for different problems. Furthermore, Agnetis *et al*. (2009) addressed different bounding schemes for the same two-agent scheduling problems.

Many papers published considered the two agents model in machine scheduling environment (Agnetis, Pacciarelli *et al*. 2007, Cheng, Ng *et al*. 2008, Liman, Panwalkar *et al*. 1996, Nong, Cheng *et al*. 2011, Wan and Yen 2009, Yin, Wu *et al*. 2012a). Baker *et al*. (2003) analyzed and examined the implications of scheduling objective function and three basic scheduling criteria, makespan, maximum lateness and total weighted completion time. Li and Yuan (2012) studied algorithms for solving objective functions in which job families are incompatible or compatible. Many papers introduced an aging effect and a learning effect in their paper (Mosheiov 2001, Liu, Tang *et al*. 2010; Cheng *et al*., 2011). Yin *et al*. (2012b) considered release dates in their paper and objective to minimize the total tardiness. They further considered minimizing the earliness penalties as the objective. Number of papers considered impacts from pre-emptive jobs, such as papers written by Wan *et al*. (2010) and Leung *et al*. (2010) and they allow for pre-emptions. The single machine scheduling problem with objective to minimize total completion time of jobs from agent A subject to an upper bound on makespan of agent B is considered by Gajpal *et al*. (2014) and Gajpal and Sahu (2014).

Problem Description

In our study, two agents (agent A and B) are processed by a common single machine. Each agent has job sets J^A and J^B for agents A and B respectively. Job set $J^A = \{J_1^A, J_2^A, J_3^A \dots\dots J_{n_A}^A\}$ consists of n_A jobs from agent A and job set $J^B = \{J_1^B, J_2^B, J_3^B \dots\dots J_{n_B}^B\}$ consists of n_B jobs from agent B. Without loss of generality we assume that two job sets have been sorted in non-decreasing order of their processing time. Thus, J_1^A/J_1^B represents the job which has the shortest processing time among the jobs of agent A/B, and $J_{n_A}^A/J_{n_B}^B$ represents the job which has the longest processing time for agent A/B.

The processing time of job J_l^A and J_m^B is denoted by p_l^A, $l = 1\dots, n_A$, and p_m^B , $m = 1\dots, n_B$ respectively. We can denote the completion time of job J_i^A from agent A and job J_m^B from agent B by $C_l^A(\sigma)$ and $C_m^B(\sigma)$ respectively, for a given job sequence σ. The objective of agent A can be denoted by $f^A(\sigma)$ and the objective of agent B can be denoted by $f^B(\sigma)$. The objective function (the total completion time of jobs in agent A) can be defined as: $f^A(\sigma) = \sum_{i=1}^{n_A} C_i^A(\sigma)$. The objective for agent B is minimizing the total completion time and it can be denoted as: $f^B(\sigma) = \sum_{j=1}^{n_B} C_j^B(\sigma)$.

The overall objective function of the considered problem is to minimize the total completion time $f^A(\sigma)$ of agent A, subject to an upper bound Q on the maximum total completion time $f^B(\sigma)$ of agent B. Based on previous literature, the problem in this paper can be denoted as: $1 \mid \sum_{j=1}^{n_B} C_j^B(\sigma) \le Q \mid \sum_{i=1}^{n_A} C_i^A(\sigma)$. The value of Q is a fixed number. Although this problem has been proved to be an NP-hard, this problem can be solved in pseudo polynomial time. We would like to investigate what is the maximum size of problem that can be solved by the exact algorithm.

The Exact Algorithm

The pseudo polynomial time algorithm was introduced by Agnetis *et al*. (2004) to solve the problem. However, there were no numerical results reported by them using this algorithm. We provide detailed description of the exact algorithm and some boundary condition missing from the paper of Agnetis *et al*. (2004). The following property can be easily proved for two agents scheduling problem considered in this paper:

Property: In an optimal schedule σ^*, jobs of agent A and agent B appears in the non-decreasing order of their process time.

The proof is easy and it can be obtained by simple exchange of jobs. This property is used to design exact algorithm. Let $P(l,m)$ denote the sum of the processing times of first *l* shortest jobs in agent A and first *m* shortest jobs in agent B (Agnetis, Mirchandani *et al*. 2004). The job set of *l* shortest job of agent A can be denoted by $J^l = \{J_1^A, J_2^A, \dots, J_l^A\}$. And the job set of *m* shortest job of agent B can be denoted by $J^m = \{J_1^B, J_2^B, \dots, J_m^B\}$. The term $P(l,m)$ represents "the sum of the processing times of the *l* shortest A-jobs and the *l* shortest jobs from agent B".

Let $F(l,m,q)$ represents the optimal solution for the problem considered in this paper with first *l* jobs from agent A and first *m* jobs from agent B. The range of value of q can be conditioned to [0, Q], and all of the value of q should be integer. According to Agnetis *et al*. (2004), we can use the dynamic programming formula:

$$F(l,m,q) = min\{F(l-1,m,q) + P(l-1,m) + p_l^A, F(l,m-1,q-P(l,m)\}.$$

The last job in the optimal schedule could be from agent A or agent B. Two situations need to be considered in optimal solution. First, if the last job is J_l^A, then, the completion time of J_l^A can be denoted by $P(l-1,m) + p_l^A$. This formula means the completion time of the preceding job of J_l^A plus the processing time of J_l^A. Second situation is that the last job is J_m^B.

The optimal value of the problem can be found on $F(n_A\ n_B\ Q)$. The following boundary condition is provided by Agnetis *et al*. (2004).

$$F(0,0,q) = 0, \qquad if\ q = 0;$$
$$F(l,m\ ,q) = +\infty, \qquad if\ q < 0.$$

The boundary conditions for the case when problem consists with the jobs of agent A only or the case where the problem consists with the jobs of agent B only are not provided by Agenetis *et al*. (2004). The boundary condition for the case with only B jobs can be represented as follows:

$$F(0,m,q)=\begin{cases}+\infty, & \sum_{k=1}^{m} C_k^B(\sigma) > q \\ 0, & \sum_{k=1}^{m} C_k^B(\sigma) \le q\end{cases}$$

In this case, the problem consists with the *j* job from agent B and no jobs from agent A. If $\sum_{k=1}^{m} C_k^B(\sigma) \le q$ the schedule is feasible, and the objective function $\sum C_i^A(\sigma)$ will be zero. On the other hand, if $\sum_{k=1}^{m} C_k^B(\sigma) > q$, then the solution will be infeasible and thus $F(0,m,q)$ is $+\infty$ to show that the problem is infeasible.

Another extreme case: when the problem consists with the jobs from agent A only is:

$$F(l,0,q)=\sum_{h=1}^{l} C_h^A(\sigma)$$

The term $F(l,0,q)$ represents the problem in which there is no job of agent B. In this case, $\sum C_j^B(\sigma)=0$ and thus $\sum C_j^B(\sigma)$ will always be less than q. Hence, the optimal solution for the function above should equal to the total completion time of jobs from agent A if all jobs in the sequence are coming from agent A.

The Proposed Heuristic Algorithms

We propose two heuristics which are motivated by the shortest processing time first (SPT) rule of single machine scheduling.

Heuristic 1

In this heuristic we assume that all jobs from agent B are processed together continuously and considered as a single job J^B. First the A jobs are sequenced according to SPT rule and then B jobs are inserted in this sequence such that the total completion time of B is less than Q. The detailed procedure and algorithm is illustrated as follows:

Step 1: The jobs from agent A are sorted in increasing order of processing time, to get sequence: $\sigma^A = \{J_1^A - J_2^A - J_3^A - \cdots - J_{n_A}^A\}$.

Step 2: The jobs from agent B are sorted in increasing order of processing time, to get sequence: $\sigma^B = \{J_1^B - J_2^B - J_3^B - \cdots - J_{n_B}^B\}$.

Step 3: Let TCT_B represents the total completion time of agent B for sequence σ^B. Insert σ^B into the sequence σ^A to get the final sequence σ. Let J_{prec}^A and J_{succ}^A represent the sets of jobs processed before J^B and after J^B respectively in final sequence σ. The sequence σ^B is inserted in σ^A in such a way that the total completion time of jobs in J_{prec}^A is less or equal to $(Q - TCT_B)/n_B$.

Step 4: Calculate the total completion time of jobs from agent A for sequence σ.

Heuristic 2

In this heuristic, the jobs from agent A and agent B are arranged together in non-decreasing order of their processing times. The detailed procedure and algorithm is illustrated below:

Step 1: The jobs from agent A and agent B are sorted together in non-decreasing order of processing time to build sequence σ. If solution is infeasible then go to step 2, otherwise go to step 3.

Step 2: Make the schedule feasible by moving the jobs from agent B towards the beginning of the sequence. We move the first job from agent B towards beginning, position by position, until the restricted condition ($f^B(\sigma) \le Q$) is satisfied. If this restricted condition can not be satisfied, then we move other jobs of agent B towards beginning until sequence σ becomes feasible. If feasible solution is found, go to step 4.

Step 3: Improve the objective function by moving the jobs from agent B in towards the end of the schedule direction. We move the last B-job towards the end of the schedule, position by position, until the upper bound Q for $\sum C_j^B$ is violated. If the upper bound Q is not violated, then we move other jobs of agent B in backward direction until sequence σ beomes feasible. And then, go to step 4.

Step 4: Calculate the of total completion time of jobs from agent A.

Numerical Analysis

We generated two types of data set called small data set and bigger data set. The small data set varies from 5 to 20 jobs while the big data set varies from 30 to 150 jobs. The processing time of jobs in two agents is scattered over the range [1, 25] uniformly. The processing times for all data sets are considered to be integers. In order to make our final schedule feasible, the value of Q has been set by $Q = \alpha f_{min}^{B} + (1-\alpha) f_{max}^{B}$, where α was assigned randomly between 0.4 to 0.6. This paper uses 29 problem instances to test the performance of proposed algorithm. For evaluating the performance of our heuristics, we calculated the optimal solution for our research problem as well and compared our heuristics' results with the optimal solution. The data size for problem instances used to get solved by optimal algorithm was limited to 20 jobs per agent.

The proposed algorithm is coded in C++ and was implemented on AMD Opteron 2.3 GHz with 16GB RAM. We use the absolute percentage deviation (APD) and relative percentage deviation (RPD) to evaluate the performance of proposed heuristics. The value of APD can be calculated as:

$$APD = \left\{\frac{Heuristic\ Solution - Optimal\ Solution}{Optimal\ Solution}\right\} \times 100$$

The value of RPD can be calculated as:

$$RPD = \{\frac{Heuristic\ Solution - Best\ Solution}{Best\ Solution}\} \times 100$$

In table 1, we compare results of proposed heuristics with the optimal solutions for evaluating two heuristics by using the absolute percentage deviation. The APD is mainly for evaluating the results via computing how far the heuristic solution is away from the optimal solution. In table 2, we compare two heuristics internally by using the relative percentage deviation. We denoted following notations for result reporting.
OPT: *Optimal solution for the problem*;
ABS: *Absolute value of the solution obtained by the heuristic*;
APD: *Absolute percentage deviation*;
RPD: *Relative percentage deviation*.

Table 1. Computational results for OPT and two heuristics for small data set.

n_A	n_B	Q	*OPT*		*Heuristic 1*			*Heuristic 2*		
			ABS	*CPU Time*	*ABS*	*APD*	*CPU Time*	*ABS*	*APD*	*CPU Time*
5	5	354	323	< 1	424	31.27	< 1	335	3.72	< 1
6	6	666	401	< 1	491	22.44	< 1	401	0.00	< 1
7	7	690	530	< 1	665	25.47	< 1	544	2.64	< 1
8	8	1036	680	< 1	835	22.79	< 1	688	1.18	< 1
9	9	1237	1083	< 1	1220	12.65	< 1	1089	0.55	< 1
10	10	1170	992	< 1	1309	31.96	< 1	1011	1.92	< 1
11	11	1435	1367	< 1	1572	15.00	< 1	1391	1.76	< 1
12	12	2173	1533	< 1	1935	26.22	< 1	1536	0.20	< 1
13	13	2876	1956	< 1	2321	18.66	< 1	1990	1.74	< 1
14	14	2444	1938	< 1	2392	23.43	< 1	1950	0.62	< 1
15	15	2785	2653	< 1	3175	19.68	< 1	2682	1.09	< 1

n_A	n_B	Q	*OPT*		*Heuristic 1*			*Heuristic 2*		
			ABS	*CPU Time*	*ABS*	*APD*	*CPU Time*	*ABS*	*APD*	*CPU Time*
16	16	3467	3165	< 1	3498	10.52	< 1	3199	1.07	< 1
17	17	4756	3106	< 1	3743	20.51	< 1	3136	0.97	< 1
18	18	3573	3060	< 1	3728	21.83	< 1	3101	1.34	< 1
19	19	5197	3145	< 1	3869	23.02	< 1	3176	0.99	< 1
20	20	4947	3921	< 1	4521	15.30	< 1	3994	1.86	< 1
Average			1865.81		2231.13	21.30		1888.94	1.35	

Table 1 reports the optimal solution, results of heuristics, and absolute performance deviation of proposed heuristics. We can see the comparison between the optimal solution and the result of proposed heuristics. We can solve the problem up to 20 jobs, because the pseudo polynomial time algorithm is a three-dimension array and it will be confined by the memory. The result presented in table 1 shows that the Heuristic 1 and Heuristic 2 are away from the optimal solution by 21.30% and 1.35% respectively. The performance of Heuristic 2 is much better than the Heuristic 1. Furthermore, we notice that with the increasing of the number of jobs, the value of APD of heuristic 2 does not change. With respect to the CPU time, all problem instances can be calculated in fraction of seconds. Even the optimal solution is able to solve the 20 job problem in few seconds. Our optimal algorithm could not solve bigger problem instances mainly because of the limited C++ language memory. The exact algorithm required the declaration of three dimensional array related to variables n_A, n_B and Q. Thus the 20 problem instance required the declaration of three dimensional array with 1,978,800 (i.e., 20×20×4947) memory allocation. Solving bigger than 20 job problem required more memory allocations and thus we could not solve the bigger problem.

Table 2. Computational results for Heuristic 1 and Heuristic 2 for bigger data set.

n_A	n_B		*Heuristic 1*		*Heuristic 2*		*Min*
		Q	*ABS*	*RPD*	*ABS*	*RPD*	
30	30	11868	11251	18.51	9494	0	9494
40	40	21577	16973	20.71	14061	0	14061
50	50	31319	28502	22.06	23351	0	23351
60	60	39014	43119	21.09	35609	0	35609
70	70	54705	55948	18.29	47297	0	47297
80	80	78752	66074	17.19	56383	0	56383
90	90	95990	79995	19.83	66756	0	66756
100	100	117798	97539	17.53	82994	0	82994
110	110	125491	113981	19.39	95473	0	95473
120	120	163974	134604	19.93	112234	0	112234
130	130	191333	159876	18.71	134675	0	134675
140	140	230591	160097	24.38	128711	0	128711
150	150	240405	202047	22.10	165479	0	165479
Average			90000.46	19.98	74809	0	

Table 2 shows the relative performance of proposed heuristics and compares the performance of proposed heuristic internally. We do not report optimal solutions in table 2, because exact algorithm could not solve more than 20 jobs problem instances. The table 2 shows that the performance of Heuristic 1 is poor with 19.98% RPD value while the result of Heuristic 2 is better than the result of the Heuristic 1 under all of situations. Heuristic 2 fully exploits the property of the total completion time objective function for single machine scheduling problem. In heuristic2, we not only considered the

job sorting in their own job agent, but also considered utilizing the SPT rule in the job schedule work too, which means that we sort all jobs together according to the SPT rule first and following by two-step adjustment. The first adjustment aims at ensuring the sequence feasible and the second step aims at minimizing the total completion time of first agent.

To sum up, heuristic 2 is a good heuristic for this scheduling problem according to the statistical data from two tables above. It is clear from tables 1 that the heuristic 2 is almost 19.95 % (i.e., 21.30 – 1.35) better than the performance of heuristic 1 in terms of optimal gap between heuristic and optimal solution. It is also evident from tables 1 and 2 that heuristic 2 has performed better that heuristic 1 in all problem instances. The way of sorting the jobs in agent B plays a vital role in solving this scheduling problem. We considered the jobs in agent B as a single job in Heuristic 1. However, we separated B jobs in Heuristic 2 and scheduled them to different positions of sequence. Moreover, we had a further adjustment after the primary sorting work.

Conclusions

We consider a single machine scheduling problem with two competing agents. The objective is to minimize the total completion time of the jobs from first agent subject to an upper bound on the total completion time of jobs from the second agent. We proposed two heuristics to solve this problem. These heuristics are based on the shortest processing time first rule of single machine scheduling. According to final results and comparisons between two heuristics, the performance of heuristic 2 is found to be better than the performance of heuristic 1. The numerical results for small data set showed that the heuristic 2 is only 1.35% away from the optimal solution. This result indicates the superiority of heuristic 2.

References

Agnetis, A., de Pascale, G., & Pacciarelli, D. (2009). A Lagrangian approach to single-machine scheduling problems with two competing agents. Journal of Scheduling, 12(4), 401-415.

Agnetis, A., Mirchandani, P. B., Pacciarelli, D., & Pacifici, A. (2004). Scheduling problems with two competing agents. Operations Research, 52(2), 229-242.

Agnetis, A., Pacciarelli, D., & Pacifici, A. (2007). Multi-agent single machine scheduling. Annals of Operations Research, 150(1), 3-15.

Baker, K. R., & Smith, J. C. (2003). A multiple-criterion model for machine scheduling. Journal of Scheduling, 6(1), 7-16.

Cardoen, B., Demeulemeester, E., & Beliën, J. (2010). Operating room planning and scheduling: A literature review. European Journal of Operational Research, 201(3), 921-932.

Cheng, T. E., Cheng, S. R., Wu, W. H., Hsu, P. H., & Wu, C. C. (2011). A two-agent single-machine scheduling problem with truncated sum-of-processing-times-based learning considerations. Computers & Industrial Engineering, 60(4), 534-541.

Cheng, T. E., Ng, C. T., & Yuan, J. J. (2008). Multi-agent scheduling on a single machine with max-form criteria. European Journal of Operational Research, 188(2), 603-609.

Gajpal, Y., Dua, A., & Sahu, S. N. (2014). Heuristics for single machine scheduling under competition to minimize total weighted completion time and makespan objectives. Lecture Notes in Management Science, 6, 99-105.

Gajpal Y. and Sahu S. N., (2014) "A pseudo polynomial time algorithm for a single machine scheduling problem under competition to minimize total weighted completion time and makespan objectives "Paper Proceedings of the 11th International Conference on Computational Management Science, 29-31 May, 2014.

Leung, J. Y. T., Pinedo, M., & Wan, G. (2010). Competitive two-agent scheduling and its applications. Operations Research, 58(2), 458-469.

Leung, J. Y. T., Pinedo, M., & Wan, G. (2010). Competitive two-agent scheduling and its applications. Operations Research, 58(2), 458-469.

Li, S., & Yuan, J. (2012). Unbounded parallel-batching scheduling with two competitive agents. Journal of Scheduling, 15(5), 629-640.

Liman, S. D., Panwalkar, S. S., & Thongmee, S. (1996). Determination of common due window location in a single machine scheduling problem. European Journal of Operational Research, 93(1), 68-74.

Liu, P., Tang, L., & Zhou, X. (2010). Two-agent group scheduling with deteriorating jobs on a single machine. The International Journal of Advanced Manufacturing Technology, 47(5-8), 657-664.

Mosheiov, G. (2001). Scheduling problems with a learning effect. European Journal of Operational Research, 132(3), 687-693.

Nong, Q. Q., Cheng, T. C. E., & Ng, C. T. (2011). Two-agent scheduling to minimize the total cost. European Journal of Operational Research, 215(1), 39-44.

Wan, G., & Yen, B. P. C. (2009). Single machine scheduling to minimize total weighted earliness subject to minimal number of tardy jobs. European Journal of Operational Research, 195(1), 89-97.

Wan, G., Vakati, S. R., Leung, J. Y. T., & Pinedo, M. (2010). Scheduling two agents with controllable processing times. European Journal of Operational Research, 205(3), 528-539.

Yin, Y., Cheng, S. R., & Wu, C. C. (2012b). Scheduling problems with two agents and a linear non-increasing deterioration to minimize earliness penalties. Information Sciences, 189, 282-292.

Yin, Y., Wu, W. H., Cheng, S. R., & Wu, C. C. (2012a). An investigation on a two-agent single-machine scheduling problem with unequal release dates. Computers & Operations Research, 39(12), 3062-3073.