Journal of Applied Mathematics and Physics, 2023, 11, 1693-1706
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

Two-Agent Makespan Minimization Problem on Parallel Machines

Siqi Zheng*, Zhaohui Liu

School of Mathematics, East China University of Science and Technology, Shanghai, China
Email: *Y30201300@mail.ecust.edu.cn

How to cite this paper: Zheng, S.Q. and Liu, Z.H. (2023) Two-Agent Makespan Minimization Problem on Parallel Machines. *Journal of Applied Mathematics and Physics*, **11**, 1693-1706.
https://doi.org/10.4236/jamp.2023.116110

Received: May 8, 2023
Accepted: June 27, 2023
Published: June 30, 2023

Abstract

A two-agent scheduling problem on parallel machines is considered in this paper. Our objective is to minimize the makespan for agent A, subject to an upper bound on the makespan for agent B. In this paper, we provide a new approximation algorithm called CLPT. On the one hand, we compare the performance between the CLPT algorithm and the optimal solution and find that the solution obtained by the CLPT algorithm is very close to the optimal solution. On the other hand, we design different experimental frameworks to compare the CLPT algorithm and the A-LS algorithm for a comprehensive performance evaluation. A large number of numerical simulation results show that the CLPT algorithm outperformed the A-LS algorithm.

Keywords

Parallel Machines, Makespan, Approximation Algorithm, Two-Agent, Empirical Results

1. Introduction

In recent years, management problems in which multiple agents compete on the usage of a common processing resource are receiving increasing attention in different application environments, such as the scheduling of multiple flights of different airlines on a common set of airport runways, berths and material/people movers (cranes, walkways, etc.) at a port for multiple ships, of clerical works among different "managers" in an office and of a mechanical/electrical workshop for different uses; and in different methodological fields, such as artificial intelligence, decision theory, and operations research.

In this paper, we consider an agent scheduling problem. Unlike classic scheduling problems, the jobs in agent scheduling problem belong to two or more

owners which we call agents. Each agent owns a set of jobs and the objective. Each agent has its own goal depending on the schedule of its jobs. All the agents share common resource. Zhao *et al.* [1] provided an algorithm A-LS with per formance ratio $2-\frac{1}{m}$. In this paper, we study a two-agent scheduling problem. The two agents are called agent A and B, respectively. All the jobs of agent A and agent B need to be scheduled on identical machines. Our objective is to minimize the makespan for agent A while keeping the makespan for agent B under a given value. The given value is indicated as threshold value for agent B. We only consider feasible threshold value in every instance of the problem, i.e., there always exists a schedule such that the makespan for agent B is no more than the threshold value. According to the notation introduced by Graham *et al.* [2], the problem is denoted by $P\,\|\,C_{\max}^{A}:C_{\max}^{B}\le Q$.

Agent scheduling problems were introduced by Baker *et al.* [3] and Agnetis *et al.* [4]. The former paper focused on minimizing a linear combination of the agents' criteria on single machine and presented some complex results. The latter paper mainly focused on two-agent scheduling problems, they summarized the complexity of some constrained optimization problems. Agnetis *et al.* [5] extended some of those problems into multi-agent version. Balasubramanian *et al.* [6] first studied the agent scheduling problem on parallel machines. They established an iteration SPT-LPT-SPT heuristic and a bicriteria genetic algorithm for the two-agent scheduling problem which is $P\,|\,inter\,|\left(C_{\max},\sum C^{j}\right)$, where inter means that the jobs of the different sets interfere with one another. Zhao *et al.* [7] considered two models of two-agent scheduling problem which are $Pm\,\|\,C_{\max}^{A}:C_{\max}^{B}\le Q$ and $Pm\,\|\sum C_{j}^{A}:C_{\max}^{B}\le Q$, and these two problems had been proved NP-hard. For the problem $Pm\,\|\,C_{\max}^{A}:C_{\max}^{B}\le Q$, Zhao *et al.* [1] provided an algorithm A-LS with performance ratio $2-\frac{1}{m}$. Moreover, when $m=2$, they presented an algorithm A-LPTE with performance ratio $\frac{1+\sqrt{17}}{4}\approx 1.28$. Zhao *et al.* [8] studied a multi-agent scheduling problem on two identical parallel machines which is $P2\,|\,CO\,|\,C_{\max}^{1},\cdots,C_{\max}^{g}$, where CO denotes the situation where each agent completes to finish its jobs as early as possible. They introduced algorithm MML and proved the performance ratio of algorithm MML is $(1+\frac{1}{6},2+\frac{1}{6},\cdots,g+\frac{1}{6})$, which was further shown to be tight. Gu *et al.* [9] extended the problem into m machines: $Pm\,|\,CO\,|\,C_{\max}^{1},\cdots,C_{\max}^{g}$. An algorithm LLS was proposed for the problem. They proved the agent's completion time is at most $i+\left(\frac{1}{3}-\frac{1}{3m}\right)$ times its minimum makespan, where the agent with the i-th smallest α point value is the i-th completed agent.

The rest of this paper is organized as follows. In section 2 we briefly describe an integer programming (IP) model and propose a new algorithm called CLPT. In section 3 we compare the performance between the CLPT algorithm and the optimal solution first and then compare the performance between the A-LS al-

gorithm and the CLPT algorithm. Finally, in section 4 we conclude the paper and give some fruitful directions for future work.

2. Integer Programming Model and the CLPT Algorithm

2.1. Problem Formulation

Agent A has n_A jobs, we denote the job set of agent A by $J^A=\left\{J_1^A,J_2^A,\cdots,J_{n_A}^A\right\}$. Agent B has n_B jobs, we denote the job set of agent B by $J^B=\left\{J_1^B,J_2^B,\cdots,J_{n_B}^B\right\}$. Let $n=n_A+n_B$. The processing time of J_j^A is p_j^A, $j=1,2,\cdots,n_A$ and the processing time of J_j^B is p_j^B, $j=1,2,\cdots,n_B$. The two agent are competing agents, i.e., $J^A\cap J^B=\varnothing$, $J=J^A\cup J^B$. Let $C_{\max}^A(\mathrm{CLPT})$ denote the makespan for agent A obtained by the CLPT algorithm, $C_{\max}^B(\mathrm{CLPT})$ denotes the makespan for agent B obtained by the CLPT algorithm, $C_{\max}^A(\mathrm{LS})$ denotes the makespan for agent A obtained by the A-LS algorithm and $C_{\max}^B(\mathrm{LS})$ denotes the makespan for agent B obtained by the A-LS algorithm. Let π be the optimal schedule. Let $C_{\max}^A(\pi)$ denote the optimal makespan for agent A and let $C_{\max}^B(\pi)$ denote the optimal makespan for agent B.

2.2. An Integer Programming Model for $P\|C_{\max}^A:C_{\max}^B\le Q$

In this subsection, we introduce an integer programming (IP) model which can help us obtain the optimal schedule in small-sized instances. Given an arbitrary optimal schedule π', Zhao *et al.* [7] has shown that if $C_{\max}^A(\pi')\le C_{\max}^B(\pi')$, then π' can be transformed into a new schedule π such that A-jobs are all scheduled before B-jobs on each machine without increasing the makespan for agent A or changing the makespan for agent B, vice versa. Obviously, is also an optimal schedule. Let π be the optimal schedule considered in the following analysis in this section.

The IP model can be formulated as follows:

$$
\begin{aligned}
&\min\ C_{\max}^A\\
&\text{s.t.}\quad \sum_{l=1}^{m}\sum_{i=1}^{n}x_{ij}^l=1,\ j=1,\cdots,n\\
&\qquad \sum_{j=1}^{n}x_{ij}^l\le 1,\ i=1,\cdots,n,\ l=1,\cdots,m\\
&\qquad L_l^A=\sum_{j\in A}\sum_{i=1}^{n}x_{ij}^l p_j,\ l=1,\cdots,m\\
&\qquad L_l^B=\sum_{j\in B}\sum_{i=1}^{n}x_{ij}^l p_j,\ l=1,\cdots,m\\
&L_l=\sum_{j\in A\cup B}\sum_{i=1}^{n}x_{ij}^l p_j,\ l=1,\cdots,m\\
&L_l=\max\left(L_l^{\max^A},L_l^{\max^B}\right),\ l=1,\cdots,m\\
&\left(L_l-L_l^{\max^A}\right)\left(L_l^{\max^A}-L_l^A\right)=0,\ l=1,\cdots,m\\
&\left(L_l-L_l^{\max^B}\right)\left(L_l^{\max^B}-L_l^B\right)=0,\ l=1,\cdots,m
\end{aligned}
\tag{1}
$$

$$C_{\max}^{A} \geq L_{l}^{\max^{A}},\ l=1,\cdots,m$$
$$C_{\max}^{B} \geq L_{l}^{\max^{B}},\ l=1,\cdots,m$$
$$C_{\max}^{B} \leq Q$$
$$x_{ij}^{l} \in \{0,1\},\ i=1,\cdots,n,\ j \in A \cup B,\ l=1,\cdots,m$$
$$L_{l}^{A}, L_{l}^{B}, L_{l}, C_{\max}^{A}, C_{\max}^{B}, L_{l}^{\max^{A}}, L_{l}^{\max^{B}} \geq 0$$

In the (IP) formulation, $L_{l}^{\max^{A}}$ represents the makespan of A jobs on machine l and $L_{l}^{\max^{B}}$ represents the makespan of B jobs on machine l. x_{ij}^{l} represents the i-th position of the jobs j on the machine l, thus $i=1,\cdots,n$, $j \in A \cup B$, $l=1,\cdots,m$ and the decision variable $x_{ij}^{l}=1$ when the job j is placed at the i-th position on the machine l, otherwise $x_{ij}^{l}=0$.

In the above integer programming model, the first constraint and the second constraint ensure that each job is scheduled exactly once. The third constraint L_{l}^{A} represents the sum of the total processing time of A jobs on machine l. The fourth constraint L_{l}^{B} represents the sum of the total processing time of B jobs on machine l. The fifth constraint L_{l} represents the sum of the total processing time of $A \cup B$ jobs on machine l. The sixth constraint indicates that the makespan of machine l is the largest between $L_{l}^{\max^{A}}$ and $L_{l}^{\max^{B}}$. The seventh constraint and the eighth constraint give the makespan of agent A and agent B on machine l. when $L_{l}-L_{l}^{\max^{A}} \neq 0$, there must be $L_{l} > L_{l}^{\max^{A}}$, which means that on the machine l, the agent A is completed before the agent B, in the optimal solution: $C_{\max}^{A}(\pi) \leq C_{\max}^{B}(\pi)$, so the makespan of agent A on machine l is equal to the sum of the total processing time of A jobs on machine l and the makespan of agent B on machine l is equal to the sum of the total processing time of all jobs on machine l which means that $L_{l}^{\max^{A}}=L_{l}^{A}$, $L_{l}^{\max^{B}}=L_{l}$, the seventh constraint and the eighth constraint are established; when $L_{l}-L_{l}^{\max^{B}} \neq 0$, there must be $L_{l} > L_{l}^{\max^{B}}$, which means that on the machine l, the agent B is completed before the agent A, in the optimal solution: $C_{\max}^{B}(\pi) \leq C_{\max}^{A}(\pi)$, so the makespan of agent B on machine l is equal to the sum of the total processing time of agent B on machine l and the makespan of agent A on machine l is equal to the sum of the total processing time of all jobs on the machine l, which means that $L_{l}^{\max^{A}}=L_{l}$, $L_{l}^{\max^{B}}=L_{l}^{B}$, the seventh constraint and the eighth constraint also holds. The ninth constraint states that the makespan of agent A is $C_{\max}^{A}$. The tenth constraint states that the makespan of agent B is $C_{\max}^{B}$. The eleventh constraint states that the makespan of the agent B on each machine does not exceed Q, where Q is a critical parameter. It directly affects the feasibility of the problem. If Q is small, then the problem has no feasible solution due to the hard constraint of $C_{\max}^{B} \leq Q$. Finally, the value of decision variable x_{ij}^{l} and the range of related parameters $L_{l}^{A}, L_{l}^{B}, L_{l}, C_{\max}^{A}, C_{\max}^{B}, L_{l}^{\max^{A}}, L_{l}^{\max^{B}}$ are given.

2.3. The CLPT Algorithm

To describe the algorithm formally, we will introduce some notations. The LPT (longest processing time) algorithm assigns a job with maximum processing

time to the first available machine. Let C_{LPT}^{A} denote the makespan obtained by applying LPT to agent A and let C_{LPT}^{B} denote the makespan obtained by applying LPT to agent B, X and Y represent either agent A or agent B. To clarify the definition, let's assume that X represents agent A and Y represents agent B. $X_i, i=1,2,\cdots,n$ denotes the load of A jobs assigned to the machine i. $Y_i, i=1,2,\cdots,n$ denotes the load of B jobs assigned to the machine i. $p(I)=\sum_{j\in I} p_j$ denotes the totle processing time of any job in subset I. We define two bounds Q^A and Q^B as follows.

$$Q^A = \max\left\{\frac{3m-1}{2m}\cdot\frac{1}{m}\sum_{j\in A} p_j, C_{LPT}^{A}\right\}, \tag{2}$$

$$Q^B = \max\left\{\frac{3m-1}{2m}\cdot\frac{1}{m}\sum_{j\in B} p_j, C_{LPT}^{B}, \frac{3m-1}{2m}Q\right\}. \tag{3}$$

Algorithm 1 and algorithm 2 for problem $P\|C_{\max}^{A}:C_{\max}^{B}\le Q$ are described as follows.

Algorithm 1 A1(X,Y)

Let $X_i = Y_i = \emptyset$ for $i = 1,2,\ldots,m$.
for $j = 1,2,\ldots,n$ **do**
  Determine the index s such that $p(X_s \cup Y_s) = \min_{1\le i\le m} p(X_i \cup Y_i)$;
  if $j \in Y$ **then**
    $Y_s = Y_s \cup \{j\}$, $l_s = j$;
  end if
  if $j \in X$ and $p(X_s) + p_j \le Q^X$ **then**
    $X_s = X_s \cup \{j\}$, $l_s = j$;
  end if
  if $j \in X$ and $p(X_s) + p_j > Q^X$ **then**
    Let $p(X_r \cup Y_r) = \min\{p(X_i \cup Y_i) | p(X_i) + p_j \le Q^X\}$, and k be the smallest job in Y_r;
    Let $X_r = X_r \cup \{j\}$ and $Y_r = Y_r \setminus \{k\}$;
    If $l_r = k$, reset $l_r = j$;
    if $p(X_s \cup Y_s) < p(X_r \cup Y_r)$ **then**
      $Y_s = Y_s \cup \{k\}$, $l_s = k$;
    else
      $Y_r = Y_r \cup \{k\}$, $l_r = k$;
    end if
  end if
end for

Algorithm 2 CLPT

Run Algorithm 1 CLPT(A,B) to generate a schedule π_{AB}.
Run Algorithm 1 CLPT(B,A) to generate a schedule π_{BA}.
If $C_{\max}^{B}(\pi_{AB}) \le \frac{3m-1}{2m}Q$, then return π_{AB}, else return π_{BA}.

In Algorithm 1 there may be more than one machine each of which has the smallest load. A1 (X, Y) always chooses the one with the minimum index.

Algorithm 1 takes $O(mn^2)$ times. Therefore, the complexity of the CLPT

algorithm is $O\left(mn^2\right)$. In order to demonstrate the feasibility of the CLPT algorithm, we propose the following lemmas.

Lemma 1. In Algorithm 1, if $j \in X$ and $p\left(X_s\right)+p_j > Q^X$, then $Y_i \neq \varnothing$ for any i with $p\left(X_i\right)+p_j \leq Q^X$.

Proof. Since $p\left(X_s \cup Y_s\right) = \min_{1\leq i\leq m} p\left(X_i \cup Y_i\right)$, we have

$$p\left(X_i \cup Y_i\right)+p_j \geq p\left(X_s \cup Y_s\right)+p_j > Q^X.$$

Then, $Y_i \neq \varnothing$.□

Lemma 2. For any $j \in X$, Algorithm 1 can always find X_r such that $p\left(X_r\right)+p_j \leq Q^X$.

Proof. Suppose to the contrary that $j \in X$ is the first job that cannot be assigned to any X_i in Algorithm 1. Then, after jobs $1,2,\cdots,j-1$ are assigned, it holds that

$$p\left(X_i\right)+p_j > Q^X, \quad i=1,2,\cdots,m. \tag{4}$$

Denote $I = X \cap \{1,2,\cdots,j-1\} = \bigcup_{i=1}^{m} X_i$. We have

$$\begin{aligned} p(I)+mp_j &= \sum_{i=1}^{m} p\left(X_i\right)+mp_j \\ &> mQ^X \\ &\geq \frac{3m}{2m+1} p(X) \\ &\geq \frac{3m}{2m+1} p(I)+\frac{3m}{2m+1} p_j. \end{aligned}$$

Equivalently, it holds that

$$2p_j > \frac{p(I)}{m} = \frac{1}{m}\sum_{i=1}^{m} p\left(X_i\right),$$

which implies that some X_is have at most one job of I, and no X_is have three or more jobs. Since $Q^X \geq \max_{i\in I} p_i$, (4) implies that each X_i contains at least one job of I. Without loss of generality, we assume that $X_1, X_2, \cdots, X_u$ each have one, and $X_{u+1}, X_{u+2}, \cdots, X_m$ each have two jobs of I. Since job j is not longer than any job of I, the jobs in $X_1, X_2, \cdots, X_u$ must be longer than those in $X_{u+1}, X_{u+2}, \cdots, X_m$. Otherwise, job j can be assigned to some X_i among $X_1, X_2, \cdots, X_u$.

Now consider the situation applying LPT to the job set X. We can schedule the job subset I such that M_i processes the job of X_i for $i=1,2,\cdots,u$, and $M_{u+1}, M_{u+2}, \cdots, M_m$ each process two jobs of $X_{u+1}, X_{u+2}, \cdots, X_m$. Then, job j cannot be finished by time Q^X in LPT, a contradiction with $Q^X \geq C_{LPT}^X$.□

3. Numerical Simulation

3.1. Compare the Performance between the CLPT Algorithm and the Optimal Solution

To illustrate the effectiveness of the CLPT algorithm, we randomly generate a small number of problems to compare the performance between the CLPT algo-

rithm and the optimal solution. We define the number of machines m, the total jobs of two agents n, the range of the maximum and minimum processing time of randomly generated jobs Δ and the value of constraint Q, which is an upper bound for the makespan for agent B and we use α to express, if $Q=(aLB,bLB)$, where

$$LB=\max\left\{\frac{p(B)}{m},\max\left(p_j^B\right)\right\}$$

then $\alpha=(a,b)$. Define $Pr=\frac{C_{\max}^A(\mathrm{CLPT})}{C_{\max}^A(\pi)}$. The experimental parameters followed by the average $\overline{Pr}$ for 100 replications for the integer programming model and the CLPT algorithm. The results are as follows:

In Table 1: the number of machines is set at three levels: 3, 4 and 5, the total jobs of two agents is set at three levels: $5m$, $10m$ and $15m$, the constraint of α is set at three levels (1, 1.2), (1.2, 1.5) and (1.5, 1.8) and the variation range of the maximum and minimum value of the randomly generated jobs processing time is set at three levels: (1, 5), (1, 10) and (1, 20).

It can be seen from the experimental results that when m, n, Δ are fixed, the average value of Pr increases with the increase of α. When m, n, α are fixed, the average value of Pr decreases with the increase of Δ; when m, α, Δ are fixed, the average value of Pr decreases with the increase of n; when n, Δ, α are fixed, the average value of Pr increases with the increase of m. In a word, in each case, the average value of Pr is very close to 1, which suggests that the results obtained by the CLPT algorithm is very close to the optimal solution. The results demonstrate that the CLPT algorithm is an effective algorithm.

3.2. Compare the Performance between the CLPT Algorithm and the A-LS Algorithm

The effectiveness and efficiency of heuristic algorithms in finding minimum makespan schedules was empirically evaluated by solving a large number of

Table 1. Result for the average $\overline{Pr}$.

m	α		(1, 5)	(1, 10)	(1, 20)		(1, 5)	(1, 10)	(1, 20)		(1, 5)	(1, 10)	(1, 20)
3	(1, 1.2)	$5m$	1.024	1.02	1.015	$10m$	1.012	1.007	1.006	$15m$	1.008	1.004	1.002
	(1.2, 1.5)		1.021	1.034	1.020		1.009	1.007	1.006		1.009	1.005	1.002
	(1.5, 1.8)		1.061	1.049	1.039		1.033	1.034	1.076		1.025	1.055	1.051
4	(1, 1.2)		1.025	1.020	1.018		1.010	1.006	1.005		1.009	1.004	1.003
	(1.2, 1.5)		1.054	1.019	1.020		1.010	1.006	1.005		1.011	1.004	1.003
	(1.5, 1.8)		1.076	1.075	1.070		1.185	1.072	1.069		1.095	1.040	1.050
5	(1, 1.2)		1.024	1.020	1.022		1.014	1.010	1.005		1.008	1.006	1.003
	(1.2, 1.5)		1.038	1.033	1.015		1.012	1.007	1.021		1.008	1.004	1.004
	(1.5, 1.8)		1.154	1.118	1.075		1.131	1.170	1.070		1.085	1.043	1.048

problems. In this section, we mainly describe the experimental framework used and the computational results.

3.2.1. A-LS Algorithm

The LS (list-scheduling) rule is: whenever a machine becomes available, the algorithm iteratively assigns a list of jobs to the least loaded machine. Let we briefly review the A−LS algorithm proposed by Zhao and Lu [1]

Algorithm 3 A−LS

1: Schedule the jobs in J_A according to the LS rule, then schedule the jobs in J_B according to the LS rule. Denote the schedule by σ_1^{LS}
2: Schedule the jobs in J_B according to the LS rule, then schedule the jobs in J_A according to the LS rule. Denote the schedule by σ_2^{LS}
3: If $C_{\max}^B(\sigma_1^{LS}) \le (2-\frac{1}{m})Q$, then return σ_1^{LS}, else return σ_2^{LS}.

3.2.2. Experimental Framework

In order to compare the relative performance between the A-LS algorithm and the CLPT algorithm, we design three experimental frameworks. In each experiment, we investigate four variables: the number of machines m, the total jobs of two agents n, the value of constraint Q and the range of the maximum and minimum processing time of randomly generated jobs Δ. In order to investigate the effect of the investigated factors on the performance of the two algorithms, we use the control variable method to design experiments. For each fixed factors, we use Python 3.2 to generate at least 1000 instances randomly to observe the effect of the investigated factors on the results. The analysis of variance was appropriate in this case given equal samples and a large number of replications. Levene's test was used to validate the assumption of homogeneous variance. Duncan's post hoc test was used to rank the heuristics.

We define several parameters to compare the relative performance of the two algorithms. Let $\rho_1 = \dfrac{C_{\max}^A(\mathrm{CLPT})}{C_{\max}^A(\mathrm{LS})}$ where $\rho_1 \le 1$ means that the CLPT algorithm has a better performance than the A-LS algorithm, let $\rho_2 = \dfrac{C_{\max}^B(\mathrm{CLPT})}{C_{\max}^B(\mathrm{LS})}$ where $\rho_2 \le 1$ means that the CLPT algorithm violates fewer constraints than the A-LS algorithm. In order to show the relative performance between the two algorithms more intuitively. For each case, we generate 1000 instances randomly, among which:

- N1 represents the number of $C_{\max}^A(\mathrm{CLPT}) < C_{\max}^A(\mathrm{LS})$;
- N2 represents the number of $C_{\max}^A(\mathrm{CLPT}) > C_{\max}^A(\mathrm{LS})$;
- N3 represents the number of $C_{\max}^B(\mathrm{CLPT}) < C_{\max}^B(\mathrm{LS})$;
- N4 represents the number of $C_{\max}^B(\mathrm{CLPT}) > C_{\max}^B(\mathrm{LS})$;
- N5 represents the number of $C_{\max}^A(\mathrm{CLPT}) \le C_{\max}^A(\mathrm{LS})$ and $C_{\max}^B(\mathrm{CLPT}) \le C_{\max}^B(\mathrm{LS})$;
- N6 represent the number of $C_{\max}^A(\mathrm{CLPT}) \ge C_{\max}^A(\mathrm{LS})$ and $C_{\max}^B(\mathrm{CLPT}) \ge C_{\max}^B(\mathrm{LS})$;

The above six parameters can compare the performance of the CLPT algorithm and the A-LS algorithm intuitively. A higher value of N5-N6 indicates that the performance of the CLPT algorithm is better than that of the A-LS algorithm, and a higher value of N6-N5 indicates that the performance of the A-LS algorithm is better than that of the CLPT algorithm.

Table 2 presents a summary of all three experiments. In experiment E1, the number of machines m = 3, the total number of jobs is set at two levels???5m and 20m, the variation range of the maximum and minimum value of the randomly generated jobs processing time is set at four levels: (1, 2), (1, 5), (1, 10) and (1, 20), and the constraint of α is set at three levels: (1, 1.2), (1.2, 1.5) and (1.5, 1.8). In experiment E2, the number of machines is set at three levels: 3, 5 and 10, the total number of jobs is set at three levels: 5m, 10m and 20m, the Δ is set at two levels: (1, 5) and (1, 10), and the constraint of α is set at two levels: (1, 1.2) and (1.5, 1.8). In experiment E3, the number of machines is set at three levels: 3, 5 and 10, the number of jobs is fixed at $n = 10m$, the Δ is set at three levels: (1, 5), (1, 10) and (1, 20), and the constraint of α is set at two levels: (1, 1.2) and (1.5, 1.8). Under each parameter in all experiments, we generated 1000 instances randomly and calculated the number from N1 to N6 among these 1000 instances.

3.2.3. Computational Results

Table 3 gives the results of experiment E1. Table 3 shows the case of $n = 5m$ and $n = 20m$. In case of m = 3, $n = 5m$: First, observe the effect of Δ on the results of the two algorithms. Table 3 shows that when Δ = (1, 2), α = (1, 1.8), N5-N6 $\in$ [48, 118]; when Δ = (1, 5), α = (1, 1.8), N5-N6 $\in$ [417, 429]; when Δ = (1, 10), α = (1, 1.8), N5-N6 $\in$ [516, 530]. We can find that when m, n, α are fixed, the performance of the CLPT algorithm better than the A-LS algorithm increased as Δ increased.

Then, observe the effect of α on the results of the two algorithms. Table 3 shows that when Δ = (1, 2): α = (1, 1.2), N5-N6 = 118; α = (1.2, 1.5), N5-N6 = 48; α = (1.5, 1.8), N5-N6 = 91; when Δ = (1, 5): α = (1, 1.2), N5-N6 = 420; α = (1.2, 1.5), N5-N6 = 429; α = (1.5, 1.8), N5-N6 = 417; when Δ = (1, 10): α = (1, 1.2), N5-N6 = 530; α = (1.2, 1.5), N5-N6 = 530; α = (1.5, 1.8), N5-N6 = 516; when Δ = (1, 20): α = (1, 1.2), N5-N6 = 584; α = (1.2, 1.5), N5-N6 = 560; α = (1.5, 1.8), N5-N6 = 474. We can find that when m, n, Δ are fixed, the performance of the CLPT algorithm better than the A-LS algorithm decreased as α increased on the whole.

Table 2. Summary of the experimental frameworks.

	m	n	Δ	α
E1	3	$5m$, $20m$	(1, 2), (1, 5), (1, 10), (1, 20)	(1, 1.2), (1.2, 1.5), (1.5, 1.8)
E2	3, 5, 10	$5m$, $10m$, $20m$	(1, 5), (1, 50)	(1, 1.2), (1.5, 1.8)
E3	3, 5, 10	$10m$	(1, 5), (1, 10), (1, 20)	(1.1.2), (1.5, 1.8)

Table 3. Result for experiment El.

n			$5m$						$20m$					
m	Δ	α	N_1	N_2	N_3	N_4	N_5	N_6	N_1	N_2	N_3	N_4	N_5	N_6
3	(1, 2)	(1, 1.2)	138	152	291	142	706	588	142	309	227	393	298	636
		(1.2, 1.5)	124	136	250	185	679	631	126	416	269	294	290	606
		(1.5, 1.8)	115	128	270	151	721	630	114	446	248	278	276	643
3	(1, 5)	(1, 1.2)	386	139	458	222	639	219	413	292	377	456	252	245
		(1.2, 1.5)	377	134	490	218	649	220	341	402	441	347	251	254
		(1.5, 1.8)	278	196	558	174	632	215	297	454	497	327	219	231
3	(1, 10)	(1, 1.2)	498	129	523	237	636	106	526	347	477	433	220	59
		(1.2, 1.5)	442	158	599	224	619	89	470	412	534	374	214	70
		(1.5, 1.8)	358	210	666	187	606	90	358	527	623	298	175	75
3	(1, 20)	(1, 1.2)	535	133	574	237	634	50	548	332	516	423	245	27
		(1.2, 1.5)	477	145	621	238	622	62	496	415	570	373	213	20
		(1.5, 1.8)	400	233	671	233	542	68	398	507	669	288	206	19

In case of $m = 3$, $n = 20m$: First observe the influence of Δ on the results of the two algorithms. When $\Delta = (1, 2)$, $\alpha = (1, 1.8)$, N6-N5 $\in [316, 367]$, which indicates that the A-LS algorithm outperformed the CLPT algorithm, and this advantage is very obvious. when $\Delta = (1, 5)$, $\alpha = (1, 1.8)$, N5-N6 $\in [3, 12]$; when $\Delta = (1, 10)$, $\alpha = (1, 1.8)$, N5-N6 $\in [100, 161]$; when $\Delta = (1, 20)$, $\alpha = (1, 1.8)$, N5-N6 $\in [187, 218]$. We can find that when m, n, α are fixed, the performance of the CLPT algorithm better than the A-LS algorithm increased as Δ increased, which is consistent with the results presented when $m = 3$, $n = 5m$. Then, we observe the effect of α. We can find that when m, n, Δ are fixed, the performance of the CLPT algorithm better than the A-LS algorithm decreased as α increased on the whole. which is also consistent with the results presented when $m = 3$, $n = 5m$.

Table 4 shows the results of experiment E2. Table 4 mainly observes the effect of the number of machines m and the number of jobs n on the experimental results. Since α and Δ have a great effect on the two algorithms, in order to avoid subjectivity, we set α at two levels: (1, 1.2) and (1.5, 1.8) and set Δ at two levels: (1, 5) and (1, 50).

In case of $\Delta = (1, 5)$, $\alpha = (1,1.2) \cup (1.5,1.8)$: when $m = 3$, $n = 5m$, N5-N6 $\in [380, 401]$; when $m = 3$, $n = 10m$, N5-N6 $\in [146, 220]$; when $m = 3$, $n = 20m$, N5-N6 $\in [-18, 32]$. This indicates that when m, α, Δ are fixed, the performance of the CLPT algorithm better than the A-LS algorithm decreased as n increased. In particular, when $m = 3$, $n = 20m$, $\alpha = (1.5, 1.8)$, the performance of the A-LS algorithm is better than the CLPT algorithm. When $m = 5$, $n = 5m$, N5-N6 $\in [477, 500]$; when $m = 5$, $n = 10m$, N5-N6 $\in [227, 284]$; when $m = 5$, $n = 20m$, N5-N6 $\in [66, 94]$. This indicates that when m, α, Δ are fixed, the performance of the CLPT algorithm better than the A-LS algorithm decreased as n increased,

Table 4. Result for experiment E2.

Δ			(1, 5)						(1, 50)					
m	n	α	N_1	N_2	N_3	N_4	N_5	N_6	N_1	N_2	N_3	N_4	N_5	N_6
3	$5m$	(1, 1.2)	431	134	416	233	634	233	574	142	594	277	587	29
		(1.5, 1.8)	301	192	557	216	593	213	429	242	669	230	540	43
	$10m$	(1, 1.2)	411	229	443	346	425	205	611	227	545	381	396	16
		(1.5, 1.8)	306	385	522	241	374	228	439	398	671	287	320	18
	$20m$	(1, 1.2)	420	320	400	420	261	229	608	315	492	467	219	6
		(1.5, 1.8)	289	500	526	305	195	213	418	512	661	322	167	7
5	$5m$	(1, 1.2)	494	130	488	272	599	99	574	136	562	322	548	8
		(1.5, 1.8)	291	270	719	162	568	91	411	298	726	214	492	8
	$10m$	(1, 1.2)	506	269	488	375	356	72	588	270	532	421	309	0
		(1.5, 1.8)	325	413	623	265	322	95	414	443	677	292	266	2
	$20m$	(1, 1.2)	487	362	458	452	186	92	542	399	521	448	153	0
		(1.5, 1.8)	322	552	626	304	144	78	369	567	683	298	135	0
10	$5m$	(1, 1.2)	540	139	538	325	536	16	638	112	508	391	497	0
		(1.5, 1.8)	316	264	792	144	592	18	404	324	778	160	516	0
	$10m$	(1, 1.2)	528	287	519	408	305	16	581	271	523	431	298	0
		(1.5, 1.8)	373	428	675	285	287	9	435	432	644	319	249	0
	$20m$	(1, 1.2)	530	373	491	470	157	15	553	377	502	465	158	0
		(1.5, 1.8)	335	568	688	280	152	12	373	560	665	327	113	0

which is consistent with the results presented when $m = 3$, $n = 5m$. When $m = 10$, $n = 10m$, the above results still holds. And we can also find that when n, Δ, α are fixed, the performance of the CLPT algorithm better than the A-LS algorithm increased as m increased on the whole.

In case of Δ = (1, 50), $\alpha = (1,1.2) \cup (1.5,1.8)$: when m = 3, $n = 5m$, N5-N6 ∈ [497, 558]; when $m = 3$, $n = 10m$, N5-N6 ∈ [302, 380]; when $m = 3$, $n = 20m$, N5-N6 ∈ [160, 213]. This indicates that This indicates that when m, α, Δ are fixed, the performance of the CLPT algorithm better than the A-LS algorithm decreased as n increased, which is consistent with the results presented when Δ = (1, 5), $\alpha = (1,1.2) \cup (1.5,1.8)$. When $m = 5$ and $m = 10$, the above results still holds. By comparing Δ = (1, 5) and Δ = (1, 50), it can be found that when Δ increases from (1, 5) to (1, 50), the value of N5-N6 also increases gradually, which is consistent with the results presented in Table 3. Moreover, when Δ = (1, 50), the number of machines $m \geq 5$ and the number of jobs is greater than 20m, N6 is always equal to 0, which clearly indicates that the performance of the CLPT algorithm outperformed the A-LS algorithm in this case.

Table 5 shows the results of experiment E3. Table 5 mainly observes the effect

Table 5. Result for experiment E3.

α			(1, 1.2)						(1.5, 1.8)					
m	n	Δ	N_1	N_2	N_3	N_4	N_5	N_6	N_1	N_2	N_3	N_4	N_5	N_6
3	$10m$	(1, 5)	440	210	401	348	443	226	324	350	521	270	380	213
		(1, 10)	537	224	505	361	416	63	378	382	636	253	366	79
		(1, 20)	581	229	531	363	408	28	396	404	679	267	331	29
5	$10m$	(1, 5)	456	267	510	367	366	96	316	434	626	263	303	109
		(1, 10)	561	245	511	403	352	15	387	424	679	268	308	20
		(1, 20)	555	267	575	360	373	0	427	424	650	313	263	1
10	$10m$	(1, 5)	516	278	514	398	324	18	367	455	671	287	258	16
		(1, 10)	535	291	543	409	300	0	422	429	642	319	252	1
		(1, 20)	570	282	503	443	275	0	395	460	672	294	246	0

of Δ on the experimental results. Because α has a significant effect on the experimental results, to avoid subjectivity, we set $\alpha = (1, 1.2)$ and $\alpha = (1.5, 1.8)$.

In case of $\alpha = (1, 1.2)$, $n = 10m$. When $m = 3$: $\Delta = (1, 5)$, N5-N6 = 217; $\Delta = (1, 10)$, N5-N6 = 253; $\Delta = (1, 20)$, N5-N6 = 380. When $m = 5$: $\Delta = (1, 5)$, N5-N6 = 270; $\Delta = (1, 10)$, N5-N6 = 337; $\Delta = (1, 20)$, N5-N6 = 373. When $m = 10$: $\Delta = (1, 5)$, N5-N6 = 306; $\Delta = (1, 10)$, N5-N6 = 300; $\Delta = (1, 20)$, N5-N6 = 275. We can find that when m, n, α are fixed, the range of N5-N6 increased as Δ increased on the whole and when n, Δ, α are fixed, the range of N5-N6 increased as m increased on the whole, which are consistent with the results presented in Table 3 and Table 4.

In case of $\alpha = (1.5, 1.8)$, $n = 10m$. When $m = 3$: $\Delta = (1, 5)$, N5-N6 = 167; $\Delta = (1, 10)$, N5-N6 = 287; $\Delta = (1, 20)$, N5-N6 = 302. When $m = 5$: $\Delta = (1, 5)$, N5-N6 = 194; $\Delta = (1, 10)$, N5-N6 = 288; $\Delta = (1, 20)$, N5-N6 = 262. When $m = 10$: $\Delta = (1, 5)$, N5-N6 = 242; $\Delta = (1, 10)$, N5-N6 = 251; $\Delta = (1, 20)$, N5-N6 = 246. We can find that the number of machines and Δ have the same effect on the experimental results of the two algorithms as the case of $\alpha = (1, 1.2)$, $n = 10m$. By comparing $\alpha = (1, 1.2)$ and $\alpha = (1.5, 1.8)$, we can study the effects of constraint α on the experimental results. It is found that the result of $\alpha = (1, 1.2)$ is better than that of $\alpha = (1.5, 1.8)$, which is also consistent with the results presented in Table 3 and Table 4.

3.2.4. Summary of Results

As expected, in most case, N1 ≥ N2, N3 ≥ N4, N5 ≥ N6, Which shows that the CLPT algorithm outperformed the A-LS algorithm. The data results show that among the four parameters: the number of machines m, the total job of two agents n, the constraint α and the Δ, the α and the Δ have significant effects on the results of the experiment. We can find that when m, n, α are fixed, the range of N5-N6 increased as the range of Δ increased on the whole and when n, Δ, α are fixed, the range of N5-N6 decreased as α increased on the whole.

4. Conclusions and Suggestions

This paper presents a new heuristic algorithm CLPT for the problem $P \| C_{\max}^{A} : C_{\max}^{B} \le Q$. Firstly, in order to study the relative performance between the CLPT algorithm and the optimal solution, we design the integer programming model to obtain the optimal solutions in small scale cases. The experimental results show that the result obtained by the CLPT algorithm is very close to the optimal solution. Then, we design three experimental frameworks to compare the relative performance between the A-LS algorithm and the CLPT algorithm. The experimental results show that the CLPT algorithm outperformed the A-LS algorithm.

Several issues are worthy of future investigations. Firstly, it can be seen from the experimental results of the CLPT algorithm that the makespan of agent A obtained by the CLPT algorithm is no more than $\frac{3m-1}{2m}$ times the optimal value, while the makespan for agent B is no more than $\frac{3m-1}{2m}$ times the threshold value. This provides a research direction: to prove that the performance ratio of the CLPT algorithm for this problem is $\left(\frac{3m-1}{2m}, \frac{3m-1}{2m}\right)$, which is a better performance ratio than the one currently exists. Secondly, we can extend the two-agent scheduling problem to the model of multi-agent scheduling problem. Finally, we can also consider changing the objective function of each agent, such as changing the maximum completion time to the sum of completion time, the sum of weighted completion time and a linear combination of two agents.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Zhao, K.J. and Lu, X.W. (2016) Two Approximation Algorithms for Two-Agent Scheduling on Parallel Machines to Minimize Makespan. *Journal of Combinatorial Optimization*, **31**, 260-278. https://doi.org/10.1007/s10878-014-9744-y

[2] Graham, R.L., Lawler, E.L., Lenstra, J.K. and Rinnooy Kan, A.H.G (1979) Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *Annals of Discrete Mathematics*, **5**, 287-326. https://doi.org/10.1016/S0167-5060(08)70356-X

[3] Baker, K.R. and Smith, J.C. (2003) A Multiple-Criterion Model for Machine Scheduling. *Journal of Scheduling*, **6**, 7-16. https://doi.org/10.1023/A:1022231419049

[4] Agnetis, A., Mirchandani, P.B., Pacciarelli, D. and Pacifici, A. (2004) Scheduling Problem with Two Competing Agents. *Operations Research*, **52**, 229-242. https://doi.org/10.1287/opre.1030.0092

[5] Agnetis, A., Pacciarelli, D. and Pacifici, A. (2007) Multi-Agent Single Machine Scheduling. *Annals of Operational Research*, **150**, 3-15. https://doi.org/10.1007/s10479-006-0164-y

[6] Balasubramanian, H., Fowler, J., Keha, A. and Pfund (2009) Scheduling Interfering Job Sets on Parallel Machines. *European Journal of Operational Research*, **199**, 55-67. https://doi.org/10.1016/j.ejor.2008.10.038

[7] Zhao, K.J. and Lu, X.W. (2013) Approximation Schemes for Two-Agent Scheduling on Parallel Machines. *Theoretical Computer Science*, **468**, 114-121. https://doi.org/10.1016/j.tcs.2012.11.002

[8] Zhao, K.J., Lu, X.W. and Gu, M.Z. (2016) A New Approximation Algorithm for Multi-Agent Scheduling to Minimize Makespan on Two Machines. *Journal of Scheduling*, **19**, 21-31. https://doi.org/10.1007/s10951-015-0460-y

[9] Zhao, K.J., Lu, X.W. and Gu, M.Z. (2018) An Algorithm for Multi-Agent Scheduling to Minimize the Makespan on *m* Parallel Machines. *Journal of Scheduling*, **21**, 483-492. https://doi.org/10.1007/s10951-017-0546-9