Journal of Systems Science and Information
Dec., 2015, Vol. 3, No. 6, pp. 525–537
DOI: 10.1515/JSSI-2015-0525

Uniform Parallel Machine Scheduling Problem with Controllable Delivery Times

Kai LI
School of Management, Hefei University of Technology, Hefei 230009*, China*
E-mail: *hfutlk*@139*.com*

Hui LI
School of Management, Hefei University of Technology, Hefei 230009*, China*
E-mail: 1056950844@*qq.com*

Bayi CHENG
School of Management, Hefei University of Technology, Hefei 230009*, China*
E-mail: *chengbayi@gmail.com*

Qing LUO
Nanling Branch, China Mobile Group Anhui Co., Ltd., Nanling 242400*, China*
E-mail: *luoqing@ah.chinamobile.com*

Abstract This paper considers the uniform parallel machine scheduling problem with controllable delivery times, which assumes that the delivery times of jobs are linear decreasing functions of the consumed resource. It aims to minimize the maximum completion time under the constraint that the total resource consumption does not exceed a given limit. For this NP-hard problem, we propose a resource allocation algorithm, named RAA, according to the feasible solution of the uniform parallel machine scheduling problem with fixed delivery times. It proves that RAA algorithm can obtain the optimal resource allocation scheme for any given scheduling scheme in $O(n \log n)$ time. Some algorithms based on heuristic algorithm LDT, heuristic algorithm LPDT and simulated annealing are proposed to solve the uniform parallel machine scheduling problem with controllable delivery times. The accuracy and efficiency of the proposed algorithms are tested based on those data with problem sizes varying from 40 to 200 jobs and 2 to 8 machines. The computational results indicate that the SA approach is promising and capable of solving large-scale problems in a reasonable time.

Keywords scheduling; uniform parallel machine; resource allocation; delivery times

1 Introduction

In the field of production scheduling, the tails of the jobs are a kind of common scheduling parameters, which are corresponding to some products' post-processing time in the reality, such as the cooling process of hot products, and the shaded drying process. Especially the delivery

Received May 25, 2015, accepted June 29, 2015
Supported by National Natural Science Foundation of China (71521001, 71471052, and 71202048), and the Specialized Research Fund for the Doctoral Program of Higher Education of China (20120111120013)

times are treated as the tails when the scheduling problems in the direct distribution mode. Generally, delivery time is regarded as a substitution word of tail in the field of scheduling. The scheduling problems subject to delivery times(equal to tails) are researched extensively by scholars.

At the earliest Carlier[1] considered the problem of scheduling independent jobs with tails on m identical machines to minimize the makespan. Following him, Drozdowski and Kubiak[2] considered scheduling of parallel tasks in which a linear programming was presented to find an optimal schedule for a given sequence with tails. Lancia[3] dealt with the problem of assigning a set of n jobs with tails, to either one of two unrelated parallel machines and scheduling each machine so that the makespan is minimized. Sourd and Nuijten[4] discussed scheduling problems that combine tails and deadlines or, equivalently, due delivery times and deadlines. Mauguière et al.[5] considered the problem of the representation of a set of dominant schedules by a sequence of groups of permutable jobs in a single machine problem with tails. Gharbi and Haouari[6] considered the problem of minimizing the makespan on identical parallel machines subject to delivery times. Vakhania[7] studied the problem of scheduling jobs with tails on a single machine with the objective to minimize the makespan. Haouari and Gharbi[8] investigated new lower bounds for the scheduling problem on identical parallel machines with tails. Gharbi and Haouari[9] addressed the makespan minimization for parallel identical machines subject to tails. Boxma and Zwart[10] gave an overview of recent researches on the impact of scheduling on the tail behavior of the response time of a job. Li and Yang[11] considered the uniform parallel machine scheduling problem with unequal release dates and delivery times to minimize the maximum completion time. The delivery times(or tails) are all assumed to be constant parameters in the above-mentioned classical deterministic situations, thus these could be classified as the problems of scheduling jobs with fixed delivery times.

In reality, the production efficiency can be raised by investing additional resource, such as money, energy, and catalyst. There is generally the assumption that additional resources inputs affect the release times and processing times at present achievements. Release time is another general parameter in scheduling problem. Job must be processed after its release time, so that release time is called as head. For example, Janiak[12] studied an extension of the classical single machine scheduling problem with release dates which is a positive linear decreasing function with respect to the amount of a common resource. For the same problem as studied in [12], Li et al.[13] designed a simulated annealing algorithm to obtain near-optimal solutions with high quality. Computational results show that the proposed algorithm is promising and is capable of solving large-scale problems in a reasonable amount of time. Zhang et al.[14] considered the single-machine scheduling problems with release time which is a positive and strictly decreasing function about resource consumption.

For controllable processing time problem, Wei and Wang[15] considered single-machine scheduling problems in which the processing time of a job is a function of its starting time and its resource allocation. Zhao and Tang[16] considered the single machine scheduling problems with deteriorating jobs whose processing times are a decreasing linear function of their starting time. Wang and Wang[17] studied a single-machine earliness-tardiness scheduling problem with due date assignment, in which the processing time of a job is a function of its starting

time and its resource allocation. Wang and Wang[18] considered scheduling problems with convex resource dependent processing times and deteriorating jobs, in which the processing time of a job is a function of its starting time and its convex resource allocation. Janiak and Portman[19] considered a single machine scheduling problem with job processing times dependent on continuously-divisible resource. Li et al.[20] considered the identical parallel machine problem to minimize the makespan with controllable processing times, in which the processing times are linear decreasing functions of the consumed resource. A simulated annealing algorithm was designed to obtain the near-optimal solutions with high quality. Hsu and Yang[21] analyzed unrelated parallel-machine scheduling resource allocation problems with position-dependent deteriorating jobs, in which each job processing time can be compressed through incurring an additional cost.

In many practical cases, the job tails (delivery times) would be shortened by using additional resources for increasing customer satisfaction just as release times and processing times. In such cases, each delivery time is a decision variable to be determined by the scheduler, who can take advantage of this flexibility to improve system performance. Scheduling problems with variable delivery times are very interesting both from the practical and theoretical point of view. For instance, such a problem arises in steel production, where color-coated steel sheets must be dried in baking ovens by gas. The drying time is inversely proportional to the gas flow intensity. The drying time of each color-coated steel sheet may be regarded as a tail time and its value is a linear decreasing function of the consumed gas. Although the scheduling problems with controllable release times or controllable processing times have been extensively studied, to the best of our knowledge, there is no paper on controllable delivery times.

Consider a manufacturer that has m machines for processing jobs. Some of the machines are newer models while others are older models. The machines are functionally the same; they only differ in terms of speed. So uniform machines are the parallel machines with different processing speeds. And uniform parallel machine problem is a extended version of identical parallel machine problem in which the machines have the same speed. In this paper we study a class of uniform parallel machine scheduling problem with controllable delivery times, in which the delivery times of jobs are supposed to be the linear decreasing functions of the consumed resource. For convenient demonstration, UPCD is short for "Uniform parallel scheduling problem with controllable delivery times", and UPFD short for "Uniform parallel scheduling problem with fixed delivery times".

To solve the new UPCD problem, we extend the existing algorithms of the corresponding UPFD problem firstly, and then introduce simulated annealing to obtain the solutions with higher quality. Simulated annealing algorithm, which is a global optimization algorithm based on neighborhood searching, has strong ability of jumping out of the local optimum and searching the global optimum or approximate optimum, and is irrelevant to the initial solution. Metropolis[22] first proposed the simulated annealing algorithm, which is an optimal algorithm simulating solid annealing process. Kirkpatrick et al.[23] applied the simulated annealing in the field of combinatorial optimization. Yin et al.[24] addressed a two-agent scheduling problem on a single machine where the objective is to minimize the total weighted earliness cost of all jobs, while keeping the earliness cost of one agent below or at a fixed level Q. A simulated

annealing algorithm was developed to derive the near-optimal solutions. Bank et al.[25], Nouria et al.[26] and Dai et al.[27] introduced the idea of simulated anneal to solve flow shop scheduling problems. Naderi et al.[28] designed simulated annealing algorithm for the job shop scheduling problem with sequence-dependent setup times. Shafia et al.[29] defined the train scheduling problem as a job shop scheduling problem and developed a simulated annealing algorithm to solve the problems with large-scales.

The paper is organized as follows. In Section 2, the problem description is provided. A resource allocation algorithm is proposed in Section 3. Section 4 is devoted to three algorithms for solving the considered UPCD problem. Section 5 lists a number of computational results to analyze the performance of the three algorithms. Finally, some concluding remarks are provided in Section 6.

2 Problem description

This paper considers a class of UPCD problem, in which the job's delivery times are assumed as the linear decreasing function of resources consumed. The objective is to minimize the maximum completion time under the given total resources. The job completion time equals to the sum of the finished time and the corresponding delivery time. Assume there are m machines $M_i(i = 1, 2, \cdots, m)$, the processing speed of which have constant difference, the processing speed of machine M_i is $s_i(s_i > 0)$. Given n jobs $J_j(j = 1, 2, \cdots, n)$, the corresponding basic processing time is $p_j(p_j > 0)$ if it is processed by a machine with speed 1. Job $J_j(\forall j = 1, 2, \cdots, n)$ can be processed on any machine, and if J_j is processed at machine M_i, the actual processing time is $p_{ij} = p_j/s_i$. Job $J_j(j = 1, 2, \cdots, n)$ has a post process after the completion, the time q_j of which is the linear decreasing function of resources consumed u_j, $q_j = \bar{q}_j - u_j$, in that $\bar{q}_j(\bar{q}_j \geq 0)$ is the basic delivery times of job J_j without any additional resources, and $u_j(0 \leq u_j \leq \bar{q}_j)$ is the additional resources for shorting the delivery times. This problem can be denoted as $Q_m|q_j = \bar{q}_j - u_j, \sum u_j \leq \hat{U}|C_{\max}$ with the three parameters notation, where Q_m means that the scheduling type is uniform parallel machine, $q_j = \bar{q}_j - u_j$ means that the delivery times has the linear decreasing relation with the additional resources, the objective is to minimize $C_{\max}$ under the constraint that the total resource consumption does not exceed a given limit $\hat{U}$.

Let Π be the universal set of scheduling schemes, $\pi(\pi \in \Pi)$ is a feasible scheduling scheme that can be run for $\pi = \{\pi^1, \pi^2, \cdots, \pi^m\}$, and $\pi^i(i = 1, 2, \cdots, m)$ is the sub-scheduling scheme at the machine M_i. We use $|\pi^i|$ to represent the number of the sub-scheduling scheme π^i, $\pi^i_k(0 \leq k \leq |\pi^i|)$ to be the kth job in the sub-scheduling scheme π^i, and $k = 0$ if and only if $|\pi^i| = 0$, when there are not any jobs in the sub-scheduling scheme π^i. Similarly, as U is the universal set of resource allocation schemes, $u(u \in U)$ is a feasible resource allocation scheme that can also be run for $u = \{u^1, u^2, \cdots, u^m\}$ corresponding to the scheduling scheme, and $u^i(i = 1, 2, \cdots, m)$ is the sub scheme formed by the resource allocation of each job in the sub-scheduling scheme at the machine M_i, with $u^i_k(0 \leq k \leq |\pi^i|)$ to be the additional resources allocated on the kth job in the sub-scheduling scheme π_i. Note the processing time and the delivery times of π^i_k as $p^i_k(p^i_k = p(\pi^i_k)/s_i)$ and $q^i_k(q^i_k = \overline{q^i_k} - u^i_k)$, with $p(\pi^i_k)/s_i$ and $\overline{q^i_k}$ to be the basic processing time and the delivery times of π^i_k. A feasible solution (π, u) can be described as a two-tuples made up by scheduling scheme π and its corresponding resource allocation scheme

u. Under the precondition without ambiguity, $U(\pi, u)$ shows the resources consumed by the feasible solution (π, u), that is $U(\pi, u) = \sum_{i=1}^{m} \sum_{k=0}^{|\pi^i|} u_k^i$.

Given a feasible solution (π, u), $S(\pi_k^i), c(\pi_k^i), C(\pi_k^i, u_k^i)$ are indicated as the starting time, the finished time and the completion time (the sum of the finished time and the delivery time) of the job π_k^i. Then

$$S(\pi_k^i) = \begin{cases} \sum_{r=1}^{k-1} p_r^i; & 1 < k \le |\pi_k^i| \\ 0; & k = 1 \end{cases} \tag{1}$$

$$c(\pi_k^i) = S(\pi_k^i) + p_k^i = \sum_{r=1}^{k} p_r^i; \quad 1 \le k \le |\pi_k^i| \tag{2}$$

$$C(\pi_k^i, u_k^i) = c(\pi_k^i) + q_k^i = \sum_{r=1}^{k} p_r^i + \overline{q_k^i} - u_k^i; \quad 1 \le k \le |\pi_k^i| \tag{3}$$

$$C_{\max}(\pi, u) = \max_{1 \le i \le m} \max_{1 \le k \le |\pi^i|} C(\pi_k^i, u_k^i) = \max_{1 \le i \le m} \max_{1 \le k \le |\pi^i|} \left(\sum_{r=1}^{k} p_r^i + \overline{q_k^i} - u_k^i \right) \tag{4}$$

The objective of this problem is to find π^* and its corresponding u^*, makes

$$C_{\max}(\pi^*, u^*) = \min_{\pi \in \Pi} \min_{u \in U} C_{\max}(\pi, u) \tag{5}$$

$$\text{s.t. } U(\pi^*, u^*) \le \hat{U} \tag{6}$$

3 Resource Allocation Algorithm

This section assumes $(\pi, 0)$ to be an arbitrary feasible solution of UPCD, in which any job's resource allocation amount is 0, obviously $(\pi, 0)$ is a feasible solution of the corresponding UPFD problem. Then we try to construct a resource allocation algorithm named RAA, which can improve the existing UPFD algorithm to solve the UPCD.

Algorithm RAA (Resource allocation algorithm of UPCD)

Step 1 Given a feasible solution $(\pi, 0)$ of UPCD;

Step 2 Calculate the sum of job's completion times $C(\pi, 0)$, and sequence n jobs in non-increasing order of the completion times;

Step 3 Let $j = 1, C(\pi_{n+1}, 0) = 0$;

Step 4 If $j > n$, then end; else $\triangle = C(\pi_j, 0) - C(\pi_{j+1}, 0)$;

Step 5 $q_{\min} = q_1$; for $k = 1$ to j: $q_{\min} = \min(q_{\min}, q_k)$;

Step 6 If $\triangle > q_{\min}$ and $\hat{U} \ge j \cdot q_{\min}$, then $C_{\max}(\pi, u) = C(\pi_1, 0) - q_{\min}$, return $C_{\max}(\pi, u)$, end.

Step 7 If $\triangle > q_{\min}$ and $\hat{U} < (j+1) \cdot q_{\min}$, then $C_{\max}(\pi, u) = C(\pi_1, 0) - \hat{U}/(j+1)$, return $C_{\max}(\pi, u)$, end.

Step 8 If $\triangle < q_{\min}$ and $\hat{U} < (j+1) \cdot \triangle$, then $C_{\max}(\pi, u) = C(\pi_1, 0) - \hat{U}/(j+1)$, return $C_{\max}(\pi, u)$, end.

Step 9 If $\triangle < q_{\min}$ and $\hat{U} \ge (j+1) \cdot \triangle$, then $\hat{U} = \hat{U} - (j+1) \cdot \triangle$. If $\hat{U} = 0$, then return $C_{\max}(\pi, u)$, end.

Step 10 For $k = 1$ to j: $q_k = q_k - \triangle$, $C(\pi_k, 0) = C(\pi_k, 0) - \triangle$; If $q_k = 0$, then return $C_{\max}(\pi, u)$, end; Else $j = j + 1$, go to Step 4.

Theorem 1 *For any feasible solution $(\pi, 0)$ of UPCD, suppose that the resource allocation scheme obtained by RAA algorithm is u^*, then u^* is the optimal resource allocation scheme of scheduling scheme π.*

Proof Assuming that $(\pi, 0)$ is the feasible solution of UPCD, $u^* = \{u_1, u_2, \cdots, u_n\}$ is the resource allocation scheme obtained by RAA algorithm. From the resource allocating for jobs by RAA, the completion times have $C(\pi_1^i, 0) \geq C(\pi_2^i, 0) \geq \cdots \geq C(\pi_n^i, 0)$. The property is proven by induction of n: Since $n = 1$, Property 1 is established apparently. Since $n = 2$, $\triangle = C(\pi_{n-1}^i, 0) - C(\pi_n^i, 0)$; suppose $q_{\min} = q_1$, for $k = 1$ to $n - 1$: $q_{\min} = \min(q_{\min}, q_k)$; There are three cases to be considered:

Case 1: $\hat{U} \leq (n-1)\triangle$, u^* is the optimal resource allocation scheme apparently.

Case 2: $\hat{U} \leq (n-1) \cdot q_{\min}$ and $q_{\min} \leq \triangle$, u^* is the optimal resource allocation scheme apparently.

Case 3: $\hat{U} \leq (n-1) \cdot q_{\min}$ and $q_{\min} > \triangle$, after allocating resources for scheduling scheme π using RAA algorithm, $u^* = \{u_1, u_2\}$ is a resource allocation scheme of π with $C_{\max}(\pi^*, u^*) = C(\pi_1, 0) - (u_1 + u_2)$, $u_1 = \triangle$, $u_2 = \min(U/n, q_{\min})$ where $q_{\min} = q_1$, for $k = 1$ to n: $q_{\min} = \min(q_{\min}, q_k)$.

Resources are allocated to the job with the longest completion time firstly, then the second-longest. u_1 is the maximum level of shortening of job J_1's delivery times, the same to u_2 for job J_2. Thus, $u^* = \{u_1, u_2\}$ is the optimal resource allocation scheme. Since $n = 3$, take the first two jobs as one job after one allocation, and then the situation is similar to $n = 2$. Since $n = k$, take the first j jobs as one job after one allocation, and then the situation is similar to $n = k - 1$. Therefore, the resource allocation scheme obtained by RAA algorithm is optimum.

Theorem 2 *The time complexity of resource allocation algorithm RAA is $O(n \log n)$.*

4 Algorithms for UPCD

The resource allocation algorithm RAA in the previous section can find the optimal resource allocation scheme in polynomial time for any given feasible solution $(\pi, 0)$ of UPCD; therefore, this section design algorithms for UPCD based on it. We allocate resources to the solution of LDT (largest delivery time firstly) and LPDT (largest processing and delivery time firstly) algorithm for obtaining the corresponding UPCD problem solution firstly, and then construct optimization algorithm for UPCD based on simulated annealing to obtain satisfactory solution of higher quality.

4.1 Algorithms for UPCD Based on Heuristic Rules

In solving UPFD problem, the most common heuristic rules are LDT and LPDT. [30] addressed the LDT algorithm for UPFD problem, and demonstrated that LDT algorithm is the $((m-1)s_1/\sum_{i=1}^{m} s_i + 1)$-approximation algorithm to UPFD, where s_1 is the fastest processing speed of the machines. Although the LDT rules can obtain the optimal solution for the corresponding single machine problem with fixed delivery times, but there is a clear defect in solving UPFD problem. Due to the different speed of the machine in the uniform parallel ma-

chine problem, the faster machine is usually given the priority for making full use of processing capacity of the machine. Taking the job with longer delivery times precedence in the LDT algorithm may make a short job assigned a faster machine, and a long job assigned to a slower one. While the delivery times of the two jobs only have small differences, longer job gets very big completion time on slow machine, so as to make the poor solution. In [31] we constructed a heuristic algorithm named LPDT, which gives priority to the job with the largest sum of delivery and processing time when assigning machine for jobs. It usually can get a better solution than LDT, and avoid the large completion time situation caused by assigning long job on the slow machine in LDT algorithm. Thus we realize the optimal allocation of resources based on LDT algorithm and LPDT algorithm of UPFD, and then the corresponding UPCD solving.

Algorithm LDT-RAA (LPDT-RAA)

Step 1 Sort all the jobs in the non-increasing order of delivery time in the job queue; (in LPDT-RAA, the jobs are sorted in the non-increasing order of the sum of the processing time and the corresponding delivery time);

Step 2 Put the job in the job queue on the machine with the smallest $C_{\max}$ (If there are more than one, then choose the slow machine), and delete the job from the collection;

Step 3 If the job queue is empty, then generate the solution $(\pi, 0)$, end; else go to Step 2;

Step 4 Implement RAA algorithm on the solution $(\pi, 0)$, get the resource allocation scheme u, and thus the solution (π, u) of UPCD are generated. Return (π, u) and $C_{\max}(\pi, u)$, end.

4.2 Simulated Annealing Algorithm for UPCD

Simulated annealing algorithm is based on neighborhood searching; therefore we construct the neighborhood generation methods firstly. Here we construct two neighborhood generation methods, swap neighborhood and insertion neighborhood.

A swap neighbor is obtained by exchanging the positions of a pair of selected jobs. The swap neighborhood $\mathcal{N}_1$ is the universal set of all the swap neighbors of the current schedule. Here we generate two random integer numbers $r_2, r_3 (r_2, r_3 \in [1, n]; r_2 \neq r_3)$ and the job J_{r_2} and J_{r_3} are not processed by the same machine. Exchange job J_{r_2} and J_{r_3} to generate a new sequence $(\pi', 0)$, and then a new neighborhood solution (π', u') can be obtained by RAA algorithm.

An insertion neighbor is obtained by inserting a selected job into a different position in the current schedule. The insertion neighborhood $\mathcal{N}_2$ is the universal set of all the insertion neighbors of the current solution. In the simulated annealing algorithm, we generate two random integer numbers $r_4 (r_4 \in [1, n])$ and $r_5 (r_5 \in [1, m])$, and job J_{r_4} is not in the sub-scheduling on machine M_{r_5}. Insert job J_{r_4} on the machine M_{r_5} to generate a new schedule $(\pi', 0)$, which is distributed resources for new neighborhood solution (π', u') according to RAA algorithm.

Algorithm SA for UPCD

Step 1 As dependency of SA to initial solution is not strong, then order all jobs with the rules LDT to $(\pi, 0)$, which is implemented the RAA algorithm to generate initial solution (π, u) and the sum of completion times $C_{\max}(\pi, u)$;

Step 2 Set the initial temperature $T = 100$;

Step 3 If $T \leq \varepsilon$ (where $\varepsilon = 0.001$) or there is no new solution at the same temperature, then return (π, u) and $C_{\max}(\pi, u)$,end.

Step 4 Set the iteration length L at the same temperature (where $L := n/2$);

Step 5 Generate a random number $r_1(r_1 \in [0, n])$;

Step 6 If $r_1 < 0.7$, then goto Step 7; else goto Step 8;

Step 7 Generate the new neighborhood solution $(\pi', 0)$ from swap neighborhood $\mathcal{N}_1$, and get the corresponding solution (π', u') and $C_{\max}(\pi', u')$; goto Step 9;

Step 8 Generate the new neighborhood solution $(\pi', 0)$ from insertion neighborhood $\mathcal{N}_2$, and get the corresponding solution (π', u') and $C_{\max}(\pi', u')$; goto Step 9;

Step 9 $\triangle C_{\max} := C_{\max}(\pi', u') - C_{\max}(\pi, u)$. If $\triangle C_{\max} < 0$, then $(\pi, u) := (\pi', u'), C_{\max}(\pi, u) := C_{\max}(\pi', u')$; goto Step 11;

Step 10 Generate a random number $r_6(r_6 \in [0, 1])$. If $\exp(- \triangle C_{\max}/T) > r_6$, then $(\pi, u) := (\pi', u'), C_{\max}(\pi, u) := C_{\max}(\pi', u')$;

Step 11 $L := L - 1$. If $L = 0$, then $T := \alpha T$ (where $\alpha = 0.8$) and goto Step 5; else goto Step 3.

Table 1 Experiments with shorter basic delivery times and more resources

m	n	$C_{\max}^{\mathrm{LDT-RAA}}$	$C_{\max}^{\mathrm{LPDT-RAA}}$	$C_{\max}^{\mathrm{SA}}$	Gap(SA)	Time(SA)
2	40	162.62	174.87	172.12	1.578	0.15
	80	386.50	383.37	383.37	0.000	1.28
	120	587.75	585.37	585.37	0.000	4.17
	160	813.37	812.25	812.00	0.031	9.82
	200	1005.88	1001.75	1005.00	0.025	19.67
4	40	120.13	114.50	112.38	1.852	0.14
	80	247.20	243.38	242.13	0.514	0.56
	120	379.62	371.00	369.50	0.404	4.14
	160	515.80	512.00	510.60	0.273	9.90
	200	635.37	630.80	630.25	0.087	19.65
6	40	62.50	60.50	59.00	2.480	0.12
	80	135.25	128.60	127.00	1.244	1.26
	120	197.60	194.90	193.00	0.515	4.25
	160	271.00	266.90	266.00	0.337	10.00
	200	332.40	327.80	327.40	0.122	20.25
8	40	44.75	39.88	38.00	4.714	0.11
	80	86.10	81.60	80.22	1.691	1.28
	120	126.44	123.70	122.78	0.744	4.19
	160	173.50	168.11	168.00	0.065	10.25
	200	212.22	207.89	207.67	0.106	20.77

5 Experiment Results and Analysis

In this section, we describe the numerical experiments to evaluate the algorithm proposed in the previous section. All algorithms are enabled by C++ language in BloodShed Dev-C++ 4.9.9.2. Experimental environment is the Pentium (R) 3.2 GHZ Dual-Core CPU, with 4GB memory and the Microsoft Windows XP professional operating system. All the experimental data are generated randomly by computer.

In order to make the experiment results more objective, four groups results cross combined by longer, shorter delivery times, more and little resources are given. In each experiment, UPCD problem with $2, 4, 6, 8$ machines, $40, 80, 120, 160, 200$ jobs are considered. We select various parameters for the actual production environment: 1) Set the speed s_i of machine M_i to a integer in $[1, 10]$; 2) Take the job length p_j to a integer $[1, 100]$ in randomly; 3) Longer basic delivery times $\bar{q}_j \in U(50, 100)$ and shorter basic delivery times $\bar{q}_j \in U(0, 30)$; 4) More resources $U = 3,000$ and little resources $U = 150$.

Table 2 Experiments with longer basic delivery times and more resources

m	n	$C_{\max}^{\mathrm{LDT-RAA}}$	$C_{\max}^{\mathrm{LPDT-RAA}}$	$C_{\max}^{\mathrm{SA}}$	Gap(SA)	Time(SA)
2	40	179.37	175.00	172.12	1.573	0.14
	80	386.00	383.88	383.37	0.133	1.20
	120	588.88	585.50	585.37	0.022	4.41
	160	819.12	812.75	812.00	0.093	9.81
	200	1004.38	1002.25	1005.00	0.075	19.37
4	40	114.75	115.00	112.50	1.961	0.15
	80	249.87	244.00	242.00	0.820	1.28
	120	379.00	370.38	369.50	0.237	4.10
	160	514.38	511.88	510.60	0.250	10.00
	200	636.80	631.20	630.25	0.151	20.01
6	40	63.25	63.75	58.60	8.078	0.12
	80	133.70	129.88	127.20	2.056	1.2
	120	199.40	194.80	193.00	0.924	4.12
	160	271.60	267.00	266.00	0.357	10.12
	200	336.60	328.20	327.30	0.274	20.18
8	40	40.70	48.56	37.75	7.248	0.12
	80	84.70	81.56	80.00	1.913	1.22
	120	128.77	124.67	122.80	1.500	4.12
	160	174.75	168.00	167.40	0.357	9.94
	200	213.60	208.80	207.66	0.274	20.56

In these four tables, LDT-RAA gets the objective function value based through carrying out RAA on LDT algorithm with LPDT-RAA and SA corresponding to the algorithm, and

defines Gap(SA) as follows:

$$\text{Gap(SA)} = \frac{\min\{C_{\max}^{\text{LDT-RAA}}, C_{\max}^{\text{LPDT-RAA}}\} - C_{\max}^{\text{SA}}}{\min\{C_{\max}^{\text{LDT-RAA}}, C_{\max}^{\text{LPDT-RAA}}\}} \times 100 \tag{7}$$

in which $\min\{C_{\max}^{\text{LDT-RAA}}, C_{\max}^{\text{LPDT-RAA}}\}$ is the less between objective function values of LDT-RAA and LPDT-RAA algorithm, $C_{\max}^{\text{SA}}$ is objective function values of SA algorithm, obviously the Gap(SA) is expressed as the quality's percentage of the best situation between another two algorithms' solution improved by SA, and time(SA) is the running time of SA algorithm in seconds.

Table 3 Experiments with longer basic delivery times and little resources

m	n	$C_{\max}^{\text{LDT-RAA}}$	$C_{\max}^{\text{LPDT-RAA}}$	$C_{\max}^{\text{SA}}$	Gap(SA)	Time(SA)
2	40	195.56	200.40	195.04	0.266	0.17
	80	396.28	397.50	394.71	0.396	1.28
	120	597.57	599.76	594.00	0.597	4.13
	160	823.23	822.69	819.75	0.357	10.26
	200	1011.13	1013.64	1010.23	0.089	20.17
4	40	146.52	147.97	142.75	2.573	0.15
	80	266.61	270.23	265.70	0.341	1.29
	120	392.19	398.10	389.69	0.637	4.39
	160	532.13	534.44	530.48	0.310	10.31
	200	651.60	655.44	650.15	0.222	20.84
6	40	104.94	105.14	103.03	0.876	0.12
	80	164.01	162.79	160.00	0.631	1.25
	120	224.82	228.41	223.66	0.516	4.31
	160	277.32	300.90	295.57	0.558	10.59
	200	358.39	361.32	356.49	0.530	20.12
8	40	92.36	95.13	91.96	0.433	0.10
	80	119.93	121.73	118.83	0.834	1.31
	120	160.33	162.64	159.58	0.468	4.34
	160	203.52	206.33	202.66	0.423	10.14
	200	243.27	245.57	242.07	0.493	21.11

What can be seen from the analysis of experimental data in these four tables are:

1) SA algorithm improves most on 8.078%, least on 0.000%, overall average on 0.922%. In addition to corresponding SA algorithms of 2 machines 80 jobs, 120 jobs in the Table 1 and Table 4 not improving the quality of the solution, which explains that the objective function value solved by SA algorithm is very close to the optimal solution, the rest of the SA algorithm's performance is all superior to LDT–RAA and LPDT–RAA algorithm.

2) SA algorithm improves 0.839% in Table 1, 1.431% in Table 2, 0.579% in Table 3, 0.839% in Table 4 on average. When the delivery times are longer, the average percentage improvement of SA algorithm with more resources is better than the one with little resources. When

there are sufficient resources, the average percentage improvement of SA algorithm with longer delivery times is better than shorter ones. It's because that SA algorithm can make full use of resources in the process of finding the optimal solution with larger resources, but the degree of resource utilization with shorter delivery times is lower than the longer ones for the delivery time limitation under the circs of the definite resource.

Table 4 Experiments with shorter basic delivery times and little resources

m	n	$C_{\max}^{\mathrm{LDT-RAA}}$	$C_{\max}^{\mathrm{LPDT-RAA}}$	$C_{\max}^{\mathrm{SA}}$	Gap(SA)	Time(SA)
2	40	176.63	174.88	172.12	1.578	0.15
	80	386.50	383.37	383.37	0.000	1.28
	120	587.75	585.37	585.37	0.000	4.29
	160	813.87	812.25	812.00	0.031	10.07
	200	1005.88	1001.75	1001.50	0.025	19.53
4	40	120.13	114.50	112.83	1.852	0.14
	80	247.20	243.38	242.13	0.514	0.23
	120	379.63	371.00	369.50	0.404	4.12
	160	515.80	512.00	510.60	0.273	9.92
	200	635.37	630.80	630.25	0.087	19.85
6	40	60.25	60.50	59.00	2.480	0.12
	80	135.25	128.60	127.00	1.244	1.21
	120	197.60	194.00	193.00	0.515	4.21
	160	271.00	266.90	266.00	0.337	10.26
	200	332.40	327.80	327.40	0.122	20.17
8	40	44.75	39.88	38.00	4.714	0.12
	80	86.10	81.60	80.22	1.691	1.21
	120	126.44	123.70	122.78	0.744	4.20
	160	173.50	168.11	168.00	0.065	10.26
	200	212.22	207.89	207.67	0.106	20.82

3) SA algorithm improves greater with the increase of machines with the definite jobs in Tables 1, 2, 4. The reason for this is that resources are more plentiful in Tables 1, 2, 4 and the increase of machine number would weaken the difference of jobs' completion times at some level. The difference of jobs' completion times after non-increased sorting become smaller with the increase of jobs under the same machine number and therefore SA algorithm improves lesser. In Table 3 data do not show the above rules because of the insufficient resources.

4) The corresponding objective function values of LDT-RAA, LPDT-RAA and SA algorithm in the Table 1 are fundamentally the same as the Table 4. With shorter delivery times in Tables 1 and 4, jobs processed in the machines can't be fully allocated the resources, so the same data value occurs in Tables 1 and 4.

5) As SA algorithm here can solve problem of 200 jobs in 23 seconds, the computation efficiency is acceptable and SA algorithm's solution accuracy is obviously superior to other

algorithms.

6 Conclusion

We address a class of uniform parallel machine scheduling problem with that job's delivery times is linear decreasing functions of the consumed resource. The objective is minimizing the maximum completion time based on total given resource, for whose restriction to delivery times, RAA is designed. Considering the NP-hard characteristics of this problem, we construct the simulated annealing algorithm to solve the problem for approximate optimal solution, in which exchanging neighborhood and inserting neighborhood are employed. To evaluate the quality of the solution, LDT-RAA and LPDT-RAA algorithm are presented for UPCD problem on the existing heuristic algorithm. Plenty of experimental data and analyzing results show that simulated annealing algorithm can solve 200 operation scales' problem effectively in 23 seconds, and the quality is better than the other two algorithms.

Future research will be focused on solving large-scale hybrid flow shop and flexible job shop problems. We would expect that the good performance of SA would prove to be useful to achieve this goal, but this requires further investigation.

References

[1] Carlier J. Scheduling jobs with release dates and tails on identical machines to minimize the makespan. European Journal of Operational Research, 1987, 29: 298–306.

[2] Drozdowski M, Kubiak W. Scheduling parallel tasks with sequential heads and tails. Annals of Operations Research, 1999, 90: 221–246.

[3] Lancia G. Scheduling jobs with release dates and tails on two unrelated parallel machines to minimize the makespan. European Journal of Operational Research, 2000, 120: 277–288.

[4] Sourd F, Nuijten W. Scheduling with tails and deadlines. Journal of Scheduling, 2001, 4: 105–121.

[5] Mauguière P, Billaut J C, Artigues C. Grouping jobs on a single machine with heads and tails to represent a family of dominant schedules. Paper presented at the 8th Workshop on Project Management and Scheduling, Valencia 3–5 April, 2002, 265–269.

[6] Gharbi A, Haouari M. Minimizing makespan on parallel machines subject to release dates and delivery times. Journal of Scheduling, 2002, 5: 329–355.

[7] Vakhania N. Single-machine scheduling with release times and tails. Annals of Operations Research, 2004, 129: 253–271.

[8] Haouari M, Gharbi A. Lower bounds for scheduling on identical parallel machines with heads and tails. Annals of Operations Research, 2004, 129: 187–204.

[9] Gharbi A, Haouari M. An approximate decomposition algorithm for scheduling on parallel machines with heads and tails. Computer & Operations Research, 2007, 34: 868–883.

[10] Boxma O, Zwart B. Tails in scheduling. ACM SIGMETRICS Performance Evaluation Review, 2007, 34: 13–20.

[11] Li K, Yang S L. Heuristic algorithms for scheduling on uniform parallel machines with heads and tails. Journal of Systems Engineering and Electronics, 2011, 22: 462–467.

[12] Janiak A. Single machine scheduling problem with a common deadline and resource dependent release dates. European Journal of Operational Research, 1991, 53: 317–325.

[13] Li K, Yang S L, Ren M L. Single-machine scheduling problem with resource dependent release dates to minimise total resource-consumption. International Journal of Systems Science, 2011, 40: 1811–1820.

[14] Zhang X G, Yan G L, Tang G C. Single-machine scheduling problems with release time of jobs depending on resource allocated. International Journal of Advanced Manufacturing Technology, 2011, 57: 1175–1181.

[15] Wei C M, Wang J B. Single-machine scheduling with time-and-resource-dependent processing times. Applied Mathematical Modelling, 2012, 36: 792–798.

[16] Zhao C L, Tang H Y. Single machine scheduling problems with deteriorating jobs. Applied Mathematics and Computation, 2005, 161: 865–874.

[17] Wang X Y, Wang J J. Single-machine due date assignment problem with deteriorating jobs and resource-dependent processing times. International Journal of Advanced Manufacturing Technology, 2013, 67: 255–260.

[18] Wang X R, Wang J J. Single-machine scheduling with convex resource dependent processing times and deteriorating jobs. Applied Mathematical Modelling, 2013, 37: 2388–2393.

[19] Janiak A, Portmann M C. Single machine scheduling with job ready and delivery times subject to resource constraints. IEEE International Symposium on Parallel and Distributed Processing, 2008: 1–7.

[20] Li K, Shi Y, Yang S L, et al. Parallel machine scheduling problem to minimize makespan with resource dependent processing times. Applied Soft Computing, 2011, 11: 5551–5557.

[21] Hsu C J, Yang D L. Unrelated parallel-machine scheduling with position-dependent deteriorating jobs and resource-dependent processing time. Optimization Letters, 2014, 8(2): 519–531.

[22] Metropolis N, Rosenbluth A W, Resenbluth M N, et al. Equation of state calculations by fast computing machines. Journal of Chemical Physics, 1953, 21: 1087–1092.

[23] Kirkpatrick S, Gelatt C D, Vecchi M P. Optimization by simulated annealing. Science, 1983, 220: 671–690.

[24] Yin Y, Wu C C, Wu W H, et al. A branch-and-bound procedure for a single-machine earliness scheduling problem with two agents. Applied Soft Computing, 2013, 13: 1042–1054.

[25] Bank M, Ghomia S F, Jolaib F, et al. Application of particle swarm optimization and simulated annealing algorithms in flow shop scheduling problem under linear deterioration. Advances in Engineering Software, 2012, 47: 1–6.

[26] Nouria B V, Fattahia P, Ramezanian R. Hybrid firefly-simulated annealing algorithm for the flow shop problem with learning effects and flexible maintenance activities. International Journal of Production Research, 2013, 51: 3501–3515.

[27] Dai M, Tang D B, Giretc A, et al. Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. Robotics and Computer-Integrated Manufacturing, 2013, 29: 418–429.

[28] Naderi B, Fatemi Ghomi S M T, Aminnayeri M. A high performing metaheuristic for job shop scheduling with sequence-dependent setup times. Applied Soft Computing, 2010, 10: 703–710.

[29] Shafia M A, Sadjadi S J, Jamili A, et al. The periodicity and robustness in a single-track train scheduling problem. Applied Soft Computing, 2012, 12: 440–452.

[30] Koulamas C, Kyparisis G J. Scheduling on uniform parallel machines to minimize maximum lateness. Operations Research Letters, 2000, 26: 175–179.

[31] Li K, Yang S L, Ma H W. A simulated annealing approach to minimize the maximum lateness on uniform parallel machines. Mathematical and Computer Modelling, 2011, 53: 854–860.