Exact and Approximation Algorithms for the Scheduling Tasks to Minimize the Number of Processors

Natalia S. Grigoreva
St.Petersburg State University
Universitetskaj nab. 7/9,
199034 St.Petersburg, Russia
n.s.grig@gmail.com

Abstract

The multiprocessor scheduling problem is one of the classic NP-hard optimization problems. The goal of this paper is to prepare algorithms for scheduling problem where set of tasks is performed on parallel identical processors. Any task can run on any processor and each processor can perform no more than one task at a time. Preemption is not allowed. The processing time of each task is known. We study both the case in which there exist precedence constrains among tasks and the case in which each task has a release time, when it becomes available for processing, and a due dates. The time by which all tasks need to be completed (makespan) is known. We have to find where and when each task will be executed.The goal is to minimize the number of processors.

We compare the nondelay schedule, in which no processor is kept idle at a time when it could begin processing a task and an inserted idle time schedule in which a processor is kept idle at this time.

We propose some approximate algorithms and branch and bound algorithm, which produces a feasible IIT(inserted idle time) schedule for a fixed makespan and the number of processors. The algorithm may be used in a binary search mode to find the smallest number of processors. To illustrate the effectiveness of our algorithms we tested them on randomly generated set of tasks.

1 Introduction

In many applications of multiprocessor scheduling problem a set of tasks is performed on parallel identical processors. Any task can run on any processor and each processor can perform no more than one task at a time. The number of processors is known and the objective is to determine a feasible schedule S such that the maximum completion time is minimized [Brucker, 2007]. We consider two basic NP-hard [Ullman, 1975] models of multiprocessor scheduling.

In: Yu. G. Evtushenko, M. Yu. Khachay, O. V. Khamisov, Yu. A. Kochetov, V.U. Malkova, M.A. Posypkin (eds.): Proceedings of the OPTIMA-2017 Conference, Petrovac, Montenegro, 02-Oct-2017, published at http://ceur-ws.org

The problem with release times and due dates while scheduling tasks to parallel identical processors is a classical combinatorial optimization problem. Following the 3-field classification scheme proposed by [Graham et al., 1979], this problem is denoted by $P|r_j|L_{max}$.

In another multiprocessor scheduling problem precedence constructions between tasks are represented by a directed acyclic task graph $G = (U, E)$. The expression $u_i \prec u_j$ means that the task u_j may be initiated only after completion of the task u_i. The goal is to minimize the maximum completion time. For this models, we can consider two problems. In the first well-known problem the number of processors m is known and the goal is to minimize the maximum completion time. In the other one the completion time is known and the goal is to minimize the number of processors. Informally the first problem can be seen as a "dual" to the second one. We are interested in the problem where the goal is to minimize the number of processors.

A lot of research in scheduling has concentrated on the construction of nondelay schedule. A nondelay schedule has been defined by Baker [Baker, 1974] as a feasible schedule in which no processor is kept idle at a time when it could begin processing a task. An inserted idle time schedule (IIT) has been defined by J.Kanet and V.Sridharam [Kanet & Sridharan, 2000] as a feasible schedule in which a processor is kept idle at a time when it could begin processing a task.

In [Grigoreva, 2014] we considered scheduling with inserted idle time for m parallel identical processors with the objective of minimizing the makespan and proposed branch and bound algorithm for multiprocessor scheduling problem with precedence-constrained tasks.

The goal of this paper is to propose IIT schedule for two models with the objective of minimizing the number of processors. We propose an approximate IIT algorithm named MPELS/IIT (minimum processors earliest latest start/ inserted idle time) and branch and bound algorithm, which produces a feasible IIT(inserted idle time) schedule for a fixed completion time T_S and m processors. The algorithm may be used in a binary search mode to find the smallest number of processors.

2 Problems Statement

We consider a system of tasks $U = \{u_1 u_2, \ldots, u_n\}$. Each task is characterized by its execution time $t(u_i)$. Set of tasks is performed on parallel identical processors. Any task can run on any processor and each processor can perform no more than one task at a time. Task preemption is not allowed.

We consider two problem.

Problem MPRD. Each task is characterized by its execution time $t(u_i)$, its release time $r(u_i)$ and its due dates $D(u_i)$. Release time $r(u_i)$ - the time at which the task is ready for processing and due dates $D(u_i)$ specifies the time limit by which should be completed. A schedule for a task set U is the mapping of each task $u_i \in U$ to a start time $\tau(u_i)$ and a processor $num(u_i)$. In order to make the feasible schedule, it is necessary that the start time $\tau(u_i)$ of each task $u_i \in U$, satisfies the inequality

$$r(u_i) \leq \tau(u_i) \leq D(u_i) - t(u_i).$$

Length of schedule S is the quantity

$$T_S = \max\{\tau(u_i) + t(u_i) | u_i \in U\}.$$

We need determine the minimum number of identical processors that are needed in order to execute all tasks in time. Then it is clear $T_S \leq D_{\max}$ where $D_{\max} = \max\{D(u_i)\ u_i \in U\}$.

Problem MPPC. Precedence constructions between tasks are represented by a directed acyclic task graph $G = \langle U, E\rangle$. E is a set of directed arcs, an arc $e = (u_i, u_j) \in E$ if and only if $u_i \prec u_j$. The expression $u_i \prec u_j$ means that the task u_j may be initiated only after completion of the task u_i. The critical path t_{cp} is the maximal path in graph G from the initial vertex to the final vertex. We need determine the minimum number of identical processors that are needed in order to execute this set of tasks in a time not exceeding the time of the critical path t_{cp}.

For each task u, we define the earliest starting time $v_{min}(u)$ and the latest start time $v_{max}(u)$. The earliest starting time is numerically equal to the length of the maximal path in the graph G from the initial vertex to the vertex u. The latest start time$v_{max}(u)$ of task u is numerically equal to the difference between the length of the critical path t_{cp} and the length of the maximal path from the task u, to the final vertex. To schedule is feasible, it is necessary, that, for each task $u_i \in U$, start time of its execution $\tau(u_i)$ satisfies the inequalities

$$v_{min}(u_i) \leq \tau(u_i) \leq v_{max}(u_i),$$

$$\tau(u_i) + t(u_i) \leq \tau(u_j), if u_i \prec u_j.$$

Subtask of these two problems is the task of constructing a feasible schedule for the given number of processors and the given execution time. To solve this problem we apply the branch and bound method in conjunction with binary search. Branch and bound method constructs a feasible schedule S of length T for m processors. The algorithm may be used in a binary search mode to find the smallest number of processors or the smallest makespan.

First, we propose an approximate IIT algorithm named MPELS/IIT. Then by combining the MPELS/IIT algorithm and *B&B* method this paper presents BB/IIT algorithm which can find optimal solutions for MPRD and MPPC scheduling problem.

3 Lower Bound of the Number of Processors

Our target is to construct a feasible schedule for a fixed makespan C_{max} with the minimum number of processors. We set $C_{max} = t_{cp}$ or $C_{max} = D_{max}$. First we define the lower bound of the number of processors [Fernandez & Bussell, 1973]. We calculate lower bound $LB1 = \lceil \sum_{i=1}^{n} t(u_i)/C_{max} \rceil$. For problem MPRD we define for each task u_i, the earliest starting time $v_{min}(u_i) = r(u_i)$ and the latest start time $v_{max}(u_i) = D(u_i) - t(u_i)$.

We consider time intervals $[t_1, t_2] \subseteq [0, C_{max}]$.

Let $L([t_1, t_2])$ be a length of time interval $[t_1, t_2]$.

Let $M(t_1, t_2)$ be the total minimal time of tasks in time interval $[t_1, t_2]$, then

$$M(t_1, t_2) = \sum_{u_i \in U} \min\{L(x(u_i)), L(y(u_i))\},$$

where

$$x(u_i) = [v_{\min}(u_i), v_{\min}(u_i) + t(u_i)] \cap [t_1, t_2],$$

$$y(u_i) = [v_{\max}(u_i), v_{\max}(u_i) + t(u_i)] \cap [t_1, t_2].$$

Then

$$LB2 = \max\{M(t_1, t_2)/C_{max} | [t_1, t_2] \in [0, C_{max}].\}$$

Let

$$LB = \max\{LB1, LB2\}.$$

4 Approximate Algorithms

We defined the lower bound of the number of processors. We set the number of processors $smin = LB$ and construct a feasible schedule step by step.

We have to define how select a task, how select a processor. Let k tasks have been put in the schedule and partial schedule S_k have been constructed. For each task u_i, we know the earliest starting time $v_{min}(u_i)$ and the latest start time $v_{max}(u_i)$, which is a priority of task.

We know the completion time of processors $time_k[1 : smin]$. Denote $t_{min}(k) = \min\{time_k[i] | i \in 1 : smin\}$ is the earliest time of ending all the tasks included in a partial solution S_k.

Definitoin 1 *The task $u_{cr} \notin S_k$ is called the delayed task for S_k, if $v_{max}(u_{cr}) < t_{min}(k)$, where $v_{max}(u_{cr}) = \min\{v_{max}(u) | u \notin S_k\}$.*

Lemma 1 *Let delayed task u_{cr} for a partial solution S_k exists, then a partial solution S_k is unfeasible.*

4.1 Approximate Algorithm MPELS/IIT

The approximate schedule is constructed by MPELS/IIT algorithm as follows:

1. Determine the processor l_0 such as $t_{\min}(l_0) = \min\{time_k[i] | i \in 1..smin\}$.
2. Select the task u_0, such as $v_{max}(u_0) = \min\{v_{max}(u_i) | v_{\min}(u_i) - t_{\min}(k) \leq I_k, u_i \notin S_k\}$.
3. If $t_{\min}(l_0) > v_{max}(u_0)$, then $smin := smin + 1$ and $time[smin] := 0$; go to 1.

4. If $idle(u_0) = v_{min}(u_0) - t_{min}(l_0) > 0$ then select task u_1 such as
$$v_{\max}(u_1) = \min\{v_{\max}(u_i)|v_{\min}(u_i) \le t_{\min}, u_i \notin S_k\}.$$
5. if $idle(u_0) > v_{\max}(u_1) - v_{\max}(u_0))$ or $t_{\min}(k) + t(u_1) \le v_{\max}(u_0))$ then select u_1 else select u_0.
6. If $idle(u_0) > 0$, and we select u_0 then choose a task $u^* \notin S_k$, which can be executed during the idle time of the processor l_0 without increasing the start time of the task u_0, namely
$v_{max}(u^*) = \min\{v_{max}(u_i)|v_{min}(u_i) + t(u_i) \le v_{min}(u_0), u_i \notin S_k\}$.
7. If the task u^* is found, then we assign to the processor l_0 the task u^*, otherwise the task u_0.

We compare schedules which constructed by MPELS/IIT algorithm with schedules constructed by nondelay algorithm named MPELS/ND.

4.2 Approximate Algorithm MPELS/ND

The approximate schedule is constructed by MPELS/ND algorithm as follows:

1. Determine the processor l_0 such as $t_{\min}(l_0) = \min\{time_k[i]|i \in 1..smin\}$.
2. Select the task u_0, such as $v_{max}(u_0) = \min\{v_{max}(u_i)|v_{min}(u_0) \le t_{min}(l_0), u_i \notin S_k\}$.
3. If $t_{\min}(l_0) > v_{max}(u_0)$ then $smin := smin + 1$; $time[smin] := 0$, go to 1.
4. Assign the task u_0 to the processor l_0.

MPELS/ND algorithm is a list algorithm and it selects a task, which can be executed without the idle of processor. If we find the task, which can begin at time $t_{\min}(l_0)$, we set this task to processor l_0.

We can select a processor by another way. At first we select a job, then select a processor for this job.

4.3 Algorithm SPELS/IIT

1. Select the task u_0, such as $v_{max}(u_0) = \min\{v_{max}(u_i)|u_i \notin S_k\}$.
2. If $t_{\min}(l_0) > v_{max}(u_0)$ then $smin := smin + 1$ and $time[smin] := 0$, go to 1.
3. Select a processor l_1 such as $time_k[l_1] = \max\{time_k[i]|v_{min}(u_0) \le time[i] \le v_{max}(u_0), i \in 1 : smin\}$.
4. Assign the task u_0 to the processor l_1.

5 Algorithm for Constructing an Optimal Schedule

The branch and bound algorithm produces a feasible IIT schedule for a fixed makespan T_S and a fixed number of processor m. In order to optimize over m we must iterate the scheduling process over possible values of m. Let m_{opt} be minimum processors of optimal schedule. We defile interval $(a, b]$ such as $a < m_{opt} \le b$.

The preliminary step of the algorithm is a heuristic search for a good solution in order to have a good upper bound for the optimum number of processors.

The upper bound $b = mL$. Then $m_{opt} \in (a, b]$.

Select $z = \lceil (a + b)/2 \rceil$ and use branch and bound method for constructing a feasible schedule $BB(U, D, m; S)$ a feasible schedule maximum lateness z. If we find a feasible schedule then we take interval $(a, z]$, else we take interval $(z, b]$ repeat .

Algorithm $SCHEDULE(U; S_{opt}, m_{opt})$

1. Calculate $a = LB - 1$ and $b = mL$.
2. While $b - a > eps$ do
3. Set $z := \lceil (a + b)/2 \rceil$.
4. Use procedure $BB(U, C_{max}, z; S, m_S)$ for constructing a feasible schedule.
5. If we find feasible schedule S , then $S_{rec} := S$; $m_{rec} := m_S$ and set $b := m_S$, else set $a := z$.
6. $S_{opt} := S_{rec}$, $m_{opt} := m_{rec}$.

6 Branch and Bound Method for Constructing a Feasible Schedule $BB(U, T, z; S, m_S)$

The branch and bound algorithm produces a feasible IIT(inserted idle time) schedule for a fixed makespan T and the number of processors z.. In order to optimize over m we must iterate the scheduling process over possible values of m.

We proposed the formal description of the branch and bound method in [Grigoreva, 2014].

In order to make the feasible schedule, it is necessary that each task $u_i \in U$, the start time of its execution $\tau(u_i)$ satisfies the inequality

$$v_{min}(u_i) \le \tau(u_i) \le v_{max}(u_i).$$

In order to describe the branch and bound method it is necessary to determine the set of tasks that we need to add to a partial solution, the order in which task will be chosen from this set and the rules that will be used for eliminating partial solutions.

Let I be the total idle time of processors in the feasible schedule S of length T_S for m processors, then $I = z \cdot T_S - \sum_{i=1}^{n} t(u_i)$.

For a partial solution σ_k we know $idle(u_i)$— idle time of processor before start the task u_i.

At each level k will be allocated a set of tasks U_k, which we call the the ready tasks. These are tasks that need to add to a partial solution σ_{k-1}, so check all the possible continuation of the partial solutions.

Definitoin 2 *Task $u \notin \sigma_k$ is called the ready task at the level k, if $r(u)$ satisfies the inequality $r(u) - t_{min}(k) \le I - \sum_{u \in \sigma_k} idle(u_i)$.*

The main way of reducing of the exhaustive search will be the earliest possible identification unfeasible solutions. We formulated and proof the rules of deleting unfeasible solutions in [Grigoreva, 2015].

Lemma 2 *Let delayed task u_{cr} for a partial solution σ_k exists, then a partial solution σ_k is unfeasible.*

Lemma 3 *Let delayed task u_{cr} for a partial solution $\sigma_k = \sigma_{k-1} \cup u_k$ exists, then for any task u, such as $\max\{t_{min}(k-1), r(u)\} + t(u) > v_{max}(u_{cr})$ a partial solution $\sigma_{k-1} \cup u$ is unfeasible.*

Lemma 4 *Let delayed task u_{cr} for a partial solution $\sigma_k = \sigma_{k-1} \cup u_k$ exists, and $\max\{t_{min}(k-1), r(u_{cr}))\} + t(u_{cr}) > v_{max}(u_k)$ then the partial solution σ_{k-1} is unfeasible.*

Another method for determining unfeasible partial solutions based on a comparison of resource requirements of tasks and total processing power. In this case we propose to modify the algorithm for determining the interval of concentration [Fernandez & Bussell, 1973] for the complete schedule. We apply this algorithm to a partial schedule σ_k and determine its admissibility.

We consider time intervals $[t_1, t_2] \subseteq [t_{\min}(k), T_S]$. Let $MP(t_1, t_2)$ be the total time of free processors in time interval $[t_1, t_2]$ then

$$MP(t_1, t_2) = \sum_{i=1}^{m} \max\{0, (t_2 - \max\{t_1, time_k[i]\})\}.$$

For all task $u_i \notin \sigma_k$ we find minimal time of its begin: $v(u_i) = \max\{v_{min}(u_i), t_{\min}(k)\}$. Let $L([t_1, t_2])$ be a length of time interval $[t_1, t_2]$.

Let $M_k(t_1, t_2)$ be the total minimal time of tasks in time interval $[t_1, t_2]$, then

$$M_k(t_1, t_2) = \sum_{u_i \notin \sigma_k} \min\{L(x_k(u_i)), L(y(u_i))\},$$

where

$$x(u_i) = [v_{\min}(u_i), v_{\min}(u_i) + t(u_i)] \cap [t_1, t_2],$$

$$y(u_i) = [v_{\max}(u_i), v_{\max}(u_i) + t(u_i)] \cap [t_1, t_2].$$

Let

$$est(\sigma_k) = \max_{[t_1, t_2] \in [t_{\min}(k), T_S]} \{M_k(t_1, t_2) - MP(t_1, t_2).\}$$

Lemma 5 *If $est(\sigma_k) > 0$ then a partial solution σ_k is unfeasible.*

Table 1: Relative Error of the Minimum Number of Processors for Approximate Algorithms

n	MPELS/RD	SPELS/RD	MPND/RD	MPELS/PC	SPELS/PC	MPED/PC
100	0.141	0.153	0.161	0.132	0.171	0.153
100	0.152	0.174	0.213	0.141	0.231	0.142
100	0.136	0.201	0.222	0.151	0.246	0.171
300	0.158	0.197	0.198	0.173	0.137	0.156
300	0.163	0.155	0.221	0.162	0.163	0.210
300	0.175	0.186	0.193	0.210	0.165	0.221
Average	0.156	0.177	0.201	0.161	0.185	0.175

Table 2: Relative Error of the Minimum Number of Processors for MPELS/IIT Method.

n	N_{opt}	$mL \leq LB + 2$	$RT \leq 0.3$	$RT > 0.3$
100	63.1	11.2	22.5	3.1
100	61.4	12.3	22.0	4.3
100	64.2	11.5	19.1	5.2
300	65.1	10.9	20.1	4.5
300	59.8	11.6	24.4	4.2
300	62.1	12.4	20.6	5.1
Average	62.60	11.63	21.44	4.37

7 Computation Result

In this section we present numerical results for the proposed algorithms. To test the approximate $MPELS/IIT$ algorithm and BB/IIT algorithm, we conducted computational experiment. The quality of the solutions we estimated average ratio of the solution value over the lower bound of the minimum number of processors LB.

We restrict the number of iterations for a searching feasible solution by branch and bound method. If a feasible schedule S of the makespan T for m processors was not received for 20000 iterations, it was assumed that this does not exist and the number of processors m increased. This approach provided a schedule for all test problems, but whether or not the solutions obtained are exact or approximate remains an open question. Therefore, the quality of the solutions was estimated against to the lower bound of the minimum number of processors LB. We used tests from Standard Task Graph Set, which is available at http://www.kasahara.elec.waseda.ac.jp/schedule/.

Standard Task Graph Set is a kind of benchmark for evaluation of multiprocessor scheduling problem with precedence-constrained tasks, where optimal decisions are known. The each series of tests consists of 180 examples. We considered tests from Standard Task Graph Set with $n = 100$, and $n = 300$, where n is the number of tasks.

First, we tested approximate algorithms. The first column of all tables contains the number of tasks n.

In Table 1 average relative error $RT = (mL - LB)/LB$ for three algorithms are presented. Results for model with release and due dates are presented in the first, second and three columns and results for model with precedence constrained tasks are presented in fourth, fifth and sixth columns. We see from table 1 that the best approximate schedules are created by approximate algorithm MPELS.

The average relative error $RT = (mL - LB)/LB$ of schedules obtained by BB/IIT algorithm and $MPELS/IIT$ algorithm are presented in next tables.

Table 2 and table 3 shows the results for $MPELS/IIT$ algorithm and BB/IIT algorithm. The column N_{opt} shows the cases (in percents) where optimal schedules were obtained by there methods. The next column shows the number of cases (in percents) in which approximate solutions with $mL \leq LB + 2$ were obtained, but optimal solutions could not be obtained because of iterations limit. But an intermediate solution can be an optimal solution. The next two columns shows the number of cases in which $RT \in (0.05, 0.3]$ and RT greater then 0.3.

Approximate solutions with the error RT of less then 10 percent were obtained by MPEST/IIT algorithm in 90 percent of the cases tested. It is seen from Table 2 that optimal solutions were obtained for 63 percent (in average) of the cases tested. It is seen from Table 3 that optimal solutions were obtained for 68,58 percent (in average) of the cases tested.

Table 3: Relative Error of the Minimum Number of Processors for BB/IIT Method.

n	N_{opt}	$mL \leq LB + 2$	$RT \leq 0.3$	$RT > 0.3$
100	70,1	10,3	16.7	2.9
100	67.4	13.1	15.4	4,1
100	71,2	14,1	9.8	4.9
300	72,1	13,8	10.2	3.9
300	63,3	13,2	20,4	3,1
300	67.4	14,3	13,6	4.7
Average	68.58	13.19	14.35	3.88

8 Conclusion

In this work we proposed the new approximate algorithm and branch and bound method for solving the multiprocessor scheduling problem with the objective of minimizing of the number of processors. We found that the minimum number processors problem can be solved within reasonable time for moderate-size systems. With an increasing number of tasks, branch and bound method requires more time to obtain the optimal solution. Limiting the number of iterations seems justified and promising way to obtain a good approximate solution.

References

[Graham et al., 1979] Graham,J.R.,Lawner, E.L. and R. Kan.(1979). Optimization and approximation in deterministic sequencing and scheduling: A survey, *Ann. of Disc. Math. 5 (10)*, 287-326.

[Brucker, 2007] Brucker,P.(2007). *Scheduling Algorithms.*Springer,Berlin.

[Ullman, 1975] Ullman,J. (1975). NP-complete scheduling problems *J. Comp. Sys. Sci.* (171), 394-394.

[Baker, 1974] Baker,K.R.(1974.) *Introduction to Sequencing.* John Wiley & Son, New York.

[Kanet & Sridharan, 2000] Kanet,J. & Sridharan, V. (2000). Scheduling with inserted idle time:problem taxonomy and literature review, *Oper.Res 48 (1)*, 99-110.

[Grigoreva, 2014] Grigoreva,N.S.(2014). Branch and bound method for scheduling precedence constrained tasks on parallel identical processors, in *Lecture Notes in Engineering and Computer Science: Proc. of The World Congress on Engineering 2014, WCE 2014, 2-4 July, 2014*,(pp. 832–836). London, U.K.

[Grigoreva, 2015] Grigoreva,N.S.(2015). Multiprocessor Scheduling with Inserted Idle Time to Minimize the Maximum Lateness, *Proceedings of the 7th Multidisciplinary International Conference of Scheduling:Theory and Applications.* (pp.814–816). Prague,MISTA.

[Fernandez & Bussell, 1973] Fernandez,E. & Bussell,B.(1973). Bounds the number of processor and time for multiprocessor optimal schedules, *IEEE Tran. on Comp. 4 (11)*, 745-751.