Single Machine Inserted Idle Time Scheduling with Release Times and Due Dates

Natalia Grigoreva

Department of Mathematics and Mechanics, St.Petersburg State University, Russia
n.s.grig@gmail.com

Abstract. The single machine scheduling problem is considered in which each task has a release dates, a processing time and a due date. The objective is to mimimize the maximum lateness. Preemption is not allowed. Scheduling problem $1|r_j|L_{max}$ is a NP-hard problem. We define an IIT (inserted idle time) schedule as a feasible schedule in which a processor is kept idle at a time when it could begin processing an operation. We propose an approximate IIT algorithm named ELS/IIT (earliest latest start/ inserted idle time) and branch and bound algorithm, which produces a feasible IIT schedule for a fixed the maximum lateness L. In order to optimize over L we must iterate the scheduling process over possible values of L. New dominance criteria are introduced to curtail the enumeration tree. By this approach it is generally possible to eliminate most of the useless nodes generated at the lowest levels of decision tree.

1 Introduction

The problem of minimizing the maximum lateness while scheduling tasks to single processor is a classical combinatorial optimization problem. Following the 3-field classification scheme proposed by Graham *et al.* [1], this problem is denoted by $1|r_j|L_{max}$. This problem relates to the scheduling problem [2], it has many applications, and it is NP-hard [3]. The approximation algorithms for single processor scheduling problem were given by Potts[4], Hall and Shmoys [5]. This algorithms used extended Jackson's rule with some modifications. The problem is solved by the extended Jackson's rule: whenever the machine is free and one or more tasks available for processing, schedule an available task with earliest due data.

This algorithms construct an nondelay schedule. A nondelay schedule has been defined by Baker[6] 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 [7] as a feasible schedule in which a processor is kept idle at a time when it could begin processing a task. J.Kanet and V.Sridharam [7] reviewed the literature with problem setting where IIT scheduling may be required.

In [8] we considered scheduling with inserted idle time for m parallel identical processors and proposed branch and bound algorithm for multiprocessor scheduling problem with precedence-constrained tasks. In [9] we proposed the approximation IIT

In: A. Kononov et al. (eds.): DOOR 2016, Vladivostok, Russia, published at http://ceur-ws.org

algorithm for $P|r_j|L_{max}$ problem. The goal of this paper is to propose IIT schedule for $1|r_j|L_{max}$ problem. We propose an approximate IIT algorithm named ELS/IIT (earliest latest start/ inserted idle time) and branch and bound algorithm, which produces a feasible IIT(inserted idle time) schedule for a fixed maximum lateness L. The algorithm may be used in a binary search mode to find the smallest maximum lateness. A new method for evaluating partial solutions, selecting the next task and new ways of reducing the exhaustive search was designed.

We consider a system of tasks $U = \{u_1 u_2, \ldots, u_n\}$. Each task is characterized by its execution time $t(u_i)$, its release time $r(u_i)$ and its due dates $D(u_i)$. Release time $r(u_i)$ is the time at which the task is ready for processing. Due date $D(u_i)$ specifies the time limit by which the task should be completed. Set of tasks is performed on one processor. Task preemption is not allowed.

A schedule for a task set U is the mapping of each task $u_i \in U$ to a start time $\tau(u_i)$. Maximum lateness of schedule S is the quantity

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

First, we propose an approximate IIT algorithm named ELS/IIT (earliest latest start/ inserted idle time). Then by combining the ELS/IIT algorithm and $B\&B$ method this paper presents BB/IIT algorithm which can find optimal solutions for single processor scheduling problem.

2 Approximate algorithm ELS/IIT

For each task u_i, we know the earliest starting time $r(u_i)$ and the latest start time $v_{max}(u_i) = D(u_i) - t(u_i)$, which is a priority of task. Let k tasks have been put in the schedule and partial schedule S_k have been constructed.

Let be $t_{min}(k)$ the time of the termination of the processor after completion all tasks from the partial schedule S_k. The approximate schedule is constructed by ELS/IIT algorithm as follows:

1. Select the task u_0, such as $v_{max}(u_0) = \min\{v_{max}(u_i) | u_i \notin S_k\}$.
2. If $idle(u_0) = r(u_0) - t_{min}(k) > 0$ then choose a task $u^* \notin S_k$, which can be executed during the idle time of the processor without increasing the start time of the task u_0.
   Namely define the start time of task u_i as $\tau(u_i) = \max\{t_{min}(k), r(u_i)\}$ then set $d(u_i) = \tau(u_i) + t(u_i)$ and find task u^* such as
   $$v_{max}(u^*) = \min\{v_{max}(u_i) | d(u_i) \leq r(u_0), u_i \notin S_k\}.$$
3. If the task u^* is found, then we assign to the processor the task u^*, otherwise the task u_0.

Suppose that L_{opt} denotes the maximum lateness of optimal schedule, while L_{ELS} denotes the the maximum lateness when the tasks are sequenced using ELS/IIT heuristic. We are interested in seeing how much worse L_{ELS} can be compared to L_{opt}. In what follows we will prove the following worse-case bound. Let $T = \sum_{k=1}^{n} t(k)$ and $t_{min} = \min\{t(u_i) | u_i \in U\}$.

Lemma 1.
$$\frac{L_{ELS} - L_{opt}}{L_{opt} + D_{max}} \leq 1 - \frac{t_{min}}{T}.$$

Proof. Suppose that the sequence $\pi = (l_1, l_2, ..., l_n)$ is generated using ELS/IIT algorithm. In schedule π let task l_j be the task with maximum lateness, then $L_{ELS} = \tau(l_j) + t(l_j) - D(l_j)$. Then we can find the task l_i such as $L_{ELS} = r(l_i) + \sum_{k=i}^{j} t(l_k) - D(l_j)$, where $1 \leq l_i \leq l_j \leq l_n$. If there is a choice, it is assumed that l_j is as small as possible and that l_i is as large as possible. Then either task l_i is the first task in the schedule or the processor will be idle before the beginning of task l_i. Consider tasks from l_i to l_j in the sequence π. If for all $i \leq k \leq j-1$ it is true that

$$D(l_k) - t(l_k) \leq D(l_{k+1}) - t(l_{k+1})$$

then
$$D(l_i) - t(l_i) \leq D(l_j) - t(l_j).$$

Otherwise we can find k such as $i \leq k \leq j-1$ and task l_k such as

$$D(l_k) - t(l_k) > D(l_{k+1}) - t(l_{k+1}).$$

But for $k+1 \leq u \leq j-1$ we have that

$$D(l_u) - t(l_u) \leq D(l_{u+1}) - t(l_{u+1}).$$

Then $\tau(l_k) < r(l_{k+1})$ and $\tau(l_k) + t(l_k) \leq r(l_{k+1})$. Hence during the application of ELS/IIT algorithm task l_{k+1} begins at its release date $\tau(l_{k+1}) = r(l_{k+1})$, which contradicts the choice of task l_i. If $k = j-1$ and task l_j begins at its release date $\tau(l_j) = r(l_j)$ then schedule π is optimal schedule.

In either case we have from the construction of schedule that

$$D(l_i) - t(l_i) \leq D(l_j) - t(l_j).$$

On the other hand

$$L_{opt} \geq r(l_i) + t(l_i) - D(l_i) \geq r(l_i) + t(l_j) - D(l_j)$$

and
$$L_{opt} \geq T - D_{max}.$$

Then

$$L_{ELS} - L_{opt} \leq r(l_i) + \sum_{k=i}^{j} t(l_k) - D(l_j) - r(l_i) - t(l_j) + D(l_j) = \sum_{k=i}^{j-1} t(l_k).$$

Then
$$\frac{L_{ELS} - L_{opt}}{L_{opt}} \leq \frac{\sum_{k=i}^{j-1} t(l_k)}{T} \leq 1 - \frac{t_{min}}{T}.$$

To illustrate ELS heuristic we consider the following example. There are two task: $r_1 = 0; t_1 = T-1; D_1 = T-1; r_2 = 0; t_2 = 1; D_2 = 1+\delta$. Then ELS/IIT algorithm will schedule the large task first and maximum lateness $L_{ELS} = T-1-\delta$. But maximum lateness of optimal schedule $L_{opt} = 1$.

This problem can be solved using extended Jackon's rule (EDD): whenever the machine is free and one or more tasks are available for processing, schedule an available task with earliest due date. We consider examples, in which EDD algorithm builds a bad schedule, while ELS algorithm builds the optimal schedule and vice versa. For this example extended Jackson's rule (EDD heuristic) makes the optimal schedule. But if we change example: $r_1 = 0; t_1 = T-1; D_1 = T; r_2 = r; t_2 = 1; D_2 = 1$, EDD heuristic will schedule the large task first and maximum lateness $L_{EDD} = T-1$. ELS/IIT algorithm generates optimal schedule $L_{ELS} = r$ and $L_{opt} = r$;

The example 2 from [5] in table 1 demonstrates the worse-case instance for approximation algorithm B,which was proposed in [5]. There are five tasks, r_i,t_i,D_i represent the release date, processing time and due date, respectively, of task i. $L_i = \tau_i + t_i - D_i$ and $v_{max}(i) = D_i - t_i$. ELS/IIT algorithm generates the optimal schedule

Table 1. Example 2

Task	r_i	t_i	D_i	$v_{max}(i)$	τ_i	$L_i(ELS)$
1	0	Q	$2Q+2$	$Q+2$	$Q+2$	0
2	1	Q	1	$1-Q$	1	Q
3	$Q+1$	1	0	-1	$1+Q$	$Q+2$
4	$2Q+1$	Q	$2Q+1$	$Q+1$	$Q+3$	$Q+2$
5	$2Q+2$	1	$Q+1$	Q	$2Q+2$	$Q+2$

$(2,3,1,5,4)$ with the maximum lateness $L_{ELS} = Q+2$. Algorithm B [5] generates schedule $(1,2,3,4,5)$ with the maximum lateness $L_B = 2Q+2$.

3 Algorithm for constructing an optimal schedule

The branch and bound algorithm produces a feasible IIT schedule for a fixed maximum lateness L. In order to optimize over L we must iterate the scheduling process over possible values of L. Let L_{opt} be maximum lateness of optimal schedule. We defile interval $(a, b]$ such as $a < L_{opt} \le b$.

First we define the low bound of maximum lateness. We calculate two low bounds

$$LB1 = \max\{r(u_i) + t(u_i) - D(u_i) | u_i \in U\}$$

and

$$LB2 = \max\{\sum_{i=1}^{n} t(u_i) - D_{max}\}.$$

Then the low bound of maximum lateness LB is

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

The upper bound $b = \sum_{i=1}^{n} t(u_i) + r_{max} - D_{min}$ [10]. Then $L_{opt} \in (a, b]$.

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

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

1. Calculate a b.
2. While $b - a > eps$ do
3. Set $z := \lceil (a + b)/2 \rceil$.
4. We recalculate due dates $D(u_i)$ as $D^*(u_i) = D(u_i) + z$, recalculate makespan

$$D_{max} = \max\{D^*(u_i) | u_i \in U\}$$

and the latest start times $v_{max}(u_i) = D^*(u_i) - t(u_i)$.
5. Use procedure $BB(U, D^*; S, L_S)$ for constructing a feasible schedule.
6. If we find feasible schedule S , then $S_{rec} := S$; $L_{rec} := L_S$ and set $b := L_S$, else set $a := z$.
7. endwhile
8. $S_{opt} := S_{rec}$, and $L_{opt} := L_{rec}$.

4 Branch and bound method for constructing a feasible schedule $BB(U, D^*; S)$

The branch and bound algorithm produces a feasible IIT(inserted idle time) schedule for a fixed maximum lateness L. In order to optimize over L we must iterate the scheduling process over possible values of L.

For the formal description of the branch and bound method we must give a definition of partial solutions. It is convenient to represent the schedule as a permutation of tasks. For each permutation of tasks $\pi = (u_{i_1}, u_{i_2}, \ldots, u_{i_n})$, one can construct a schedule S_π as follows: the task is assigned to the processor at the earliest possible time. Partial solution σ_k, where k the number of jobs will be regarded as a partial permutation $\sigma_k = (u_{i_1}, u_{i_2}, \ldots, u_{i_k})$, which is determined partial schedule.

Definition 1. *The solution $\gamma_n = (l_1, l_2, \ldots, l_n)$ is called the extension of partial solutions $\sigma_k = (q_1, q_2, \ldots, q_k)$, if $l_1 = q_1, l_2 = q_2, \ldots, l_k = q_k$.*

Definition 2. *A partial solution σ_k is called a feasible if there exists an extension of σ_k, which is a feasible schedule.*

For each task u_i, we know the earliest starting time $r(u_i)$ and the latest start time $v_{max}(u_i) = D(u_i) - t(u_i)$, 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

$$r(u_i) \leq \tau(u_i) \leq v_{max}(u_i).$$

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 processor in the feasible schedule S of length D_{max}, then $I = D_{max} - \sum_{i=1}^{n} t(u_i)$.

For a partial solution σ_k we know for task $idle(u_i)$— 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.

Definition 3. *Task $u \notin \sigma_k$ is called the ready task at the level k, if $r(u)$ satisfies the inequality $r(u) - t_{min}(k) \leq I - \sum_{u \in \sigma_k} idle(u_i)$.*

The main way of reducing of the exhaustive search will be the earliest possible identification unfeasible solutions.

Definition 4. *Let the task $u_{cr} \notin \sigma_k$ is such as $v_{max}(u_{cr}) = \min\{v_{max}(u) | u \notin \sigma_k\}$. The task $u_{cr} \notin \sigma_k$ is called the delayed task for σ_k, if $v_{max}(u_{cr}) < t_{min}(k)$.*

Below we formulate and proof the rules of deleting unfeasible partial solutions.

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

1. *The partial solution σ_k is unfeasible.*
2. *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.*
3. *If $\max\{t_{min}(k-1), r(u_{cr})\} + t(u_{cr}) > v_{max}(u_k)$ then the partial solution σ_{k-1} is unfeasible.*

Proof. 1. This follows from definition the delayed task.

2. Let $t_{min}(k)$ is the time of ending all tasks which are included in a partial solution σ_k

If the task u_{cr} is delayed task, then $v_{\max}(u_{cr}) < t_{min}(k)$.

After cancelation of the last scheduled task u_k, algorithm returns to the partial solution σ_{k-1}. Processor ends all task at time $t_{min}(k-1)$. If we add a task u to the partial solution σ_{k-1} on step k, we must assign the task u_{cr} on the processor on step $k+1$. Therefore should be performed

$$\max\{t_{min}(k), r(u)\} + t(u) \leq v_{\max}(u_{cr}).$$

3. Consider two cases.

3.1 $v_{\max}(u_k) \leq v_{\max}(u_{cr})$. If the task u_{cr} is delayed task, then $v_{\max}(u_{cr}) < t_{min}(k)$.

After deleting the task u_k, the task u_{cr} is assigned to processor. Processor ends all it's task at time $t_{min}(k) = \max\{t_{min}(k-1), r(u_{cr})\} + t(u_{cr})$.

On lemma $\max\{t_{min}(k-1), r(u_{cr})\} + t(u_{cr}) > v_{max}(u_k)$, then the task u_k will be the delayed task for partial solution $\sigma_k = \sigma_{k-1} \cup u_{cr}$.

3.2. If $v_{\max}(u_k) > v_{\max}(u_{cr})$ then the partial solution $\sigma_{k-1} \cup u_{cr}$ was tested early and it was unfeasible. For any solution $\sigma_{k-1} \cup u$ task u_k or task u_{cr} will be delayed task. The partial solution $\sigma_k = \sigma_{k-1} \cup u$ is unfeasible for all u, then the partial solution σ_{k-1} is unfeasible.

Algorithm 1 BB/IIT algorithm

```
Set k := 1; t_min(0) := 0; σ_0 = ∅;
while (k > 0) and (k < n + 1) do
   Determine the task u_cr such as v_max(u_cr) = min{v_max(u) | u ∉ σ_{k−1}};
   if v_max(u_cr) ≤ t_min(k) then
      Compute EST = est(σ_{k−1});
      if EST ≤ 0 then
         Select the task u_0, use ELS/IIT procedure
         Set the task u_0 on processor and create partial solution σ_k = σ_{k−1} ∪ u_0
      else
         Perform step back and create the partial schedule σ_{k−1}
      else
         Delete all unfeasible partial solution by using Lemma 2
      end if
   end if
end while
if k = 0, then
   Maximum lateness of optimal schedule is greater then L_S.
end if
if k = n, then
   We find feasible schedule S = σ_n and its maximum lateness is equal L_S
end if
```

Another method for determining unfeasible partial solutions based on a comparison of resource requirements of tasks and processor power. In this case we propose to modify the algorithm for determining the interval of concentration [11] for the complete schedule. We apply this algorithm to a partial schedule σ_k and determine its admissibility.

We consider time intervals $[t_1, t_2] \subseteq [t_{\min}(k), D_{max}]$.

For all tasks $u_i \notin \sigma_k$ we find minimal time of its begin: $v(u_i) = \max\{r(u_i), time_k\}$. 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

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

Let

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

Lemma 3. *If $est(\sigma_k) > 0$ then a partial solution σ_k is unfeasible.*

The pseudo-code of Branch and bound method for constructing a feasible schedule $BB(U, D; S)$ is shown in Algorithm 1.

5 Conclusions

In this paper we propose IIT schedule for $1|r_j|L_{max}$ problem. We propose an approximate IIT algorithm named ELS/IIT (earliest latest start/ inserted idle time) and branch and bound algorithm, which produces a feasible IIT(inserted idle time) schedule for a fixed maximum lateness L. The algorithm may be used in a binary search mode to find the smallest maximum lateness. We compare IIT algorithm and algorithms which use extended Jackson's rule. We can see, that algorithms build good schedule for various examples, so combining the two approaches, we can get the best solutions for all examples.

References

1. Graham R.L., Lawner E.L. and R. Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey // Ann. of Disc. Math. 1979. Vol. 5, 10. P. 287–326.
2. P. Brucker. *Scheduling Algorithms*, (1997).
3. Lenstra J.A.,R. Kan. and Brucker P. Complexity of machine scheduling problems//Ann. of Disc. Math. 1977, 1 P.343–362.
4. Potts C.N. Analysis of a heuristic for one machine sequencing with release dates and delivery times// Operational Research. 1980. V.28 No. 6, P. 445–462.
5. Hall L.A., Shmoys D.B. Jackson's rule for single-mashine scheduling: making a good heuristic better// Mathematics of operations research. 1992. V.17., No.1. P 22–35.
6. K.R.Baker. *Introduction to Sequencing.* John Wiley & Son, New York(1974).
7. J. Kanet and V. Sridharan. Scheduling with inserted idle time:problem taxonomy and literature review, *Oper.Res* **48 (1)**, pp.99-110, (2000).
8. Grigoreva N.S. Branch and bound method for scheduling precedence constrained tasks on parallel identical processors // Lecture Notes in Engineering and Computer Science: Proc. of The World Congress on Engineering 2014, WCE 2014, 2–4 July, 2014, London, U.K., P. 832–836.
9. Grigoreva N.S. Multiprocessor Scheduling with Inserted Idle Time to Minimize the Maximum Lateness// Proceedings of the 7th Multidisciplinary International Conference of Scheduling: Theory and Applications. Prague, MISTA. 2015, P. 814–816.
10. Mastrolilli M. Efficient approximation schemes for scheduling problems with release dates and delivery times//Journal of Scheduling.2003.6,P.521–531.
11. Fernandez E., Bussell B. Bounds the number of processors and time for multiprocessor optimal schedules //IEEE Trans. on Computers. 1973, Vol. 4, 11 P. 745–751.