Faster approximation algorithms for scheduling tasks with a choice of start times

Daya Gaur * Ramesh Krishnamurti †

Abstract

We consider the problem of determining the largest independent set in a set of horizontal intervals, where two intervals are defined to be independent if they are not stabbed by any axis-parallel line. This problem arises naturally in non-preemptive scheduling of tasks on uni-processors. We study a Max SNP-hard restriction of the problem and give a randomized and a deterministic approximation algorithm. The deterministic algorithm is the first with performance ratio strictly greater than $1/2$ to the best of our knowledge.

Keywords: Approximation algorithms, Interval scheduling, Non-preemptive scheduling on single processor, Job interval selection problem.

1 Introduction

Given a set of horizontal intervals in 2D, define two intervals to be independent if their projections do not overlap neither in the x-axis nor in the y-axis. The problem is to determine the largest independent set of intervals. This problem has applications in the area of task scheduling. Let n be the minimum number of horizontal lines that stab all the intervals, and k be the maximum number of intervals that lie on any horizontal line. Each horizontal line h corresponds to a task. Intervals $(h_{i1}, h_{i2}, \ldots, h_{ij} : j \leq k)$ that lie on h correspond to the possible slots when the task can be executed. Note that the processing time of a task is variable, in that it depends on the start time. A task needs to be scheduled only once. The problem is to schedule maximum number of tasks without preemption on a single processor. Equivalently, determine the largest cardinality independent set of intervals. This problem is also known as the interval scheduling problem [7] and the job interval selection problem (JISP) [2, 4]. The problem arises in the manufacturing of printed circuit boards [8], in computation in real-time environments [6], adaptive rate controlled scheduling for multimedia applications [9] and time constrained communication scheduling [1].

*Department of Math and Computer Science, University of Lethbridge, Lethbridge, AB, Canada, T1K 3M4, gaur@cs.uleth.ca

†School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, V5A 1S6, ramesh@cs.sfu.ca

Keil [5] showed that the interval scheduling problem is NP-complete even when each task can be scheduled in at most three ($k = 3$) places. Spieksma and Crama [8] established that the decision version of the problem is strongly NP-complete even for the case when $k = 3$, and the processing times are either 1 or 2, thereby improving the result of Keil. Spieksma [7] showed that there does not exist a polynomial time approximation scheme for the problem for all $k \geq 2$ unless P = NP. Spieksma [7] also gave an approximation algorithm with performance ratio $1/2$ for the problem, and posed as an open question the design of approximation algorithms with improved performance ratio. Chuzhoy, Ostrovsky and Rabani [2] gave a randomized approximation algorithm (based on a linear programming relaxation) with an expected performance ratio of $((e-1)/e) - \epsilon$ for the general case; for the case when $k = 2$ their bound is $3/4 - \epsilon$. The need to solve a linear program motivates us to design a simpler and faster algorithm. In this paper we address this question and for the case when $k = 2$ we give a randomized approximation algorithm with a performance ratio of 0.5131057527 and a deterministic approximation algorithm with a performance ratio of $.5128269905$. This deterministic algorithm is the first with performance ratio strictly greater than $1/2$ to the best of our knowledge. In the rest of the paper, we assume there is no restriction on the processing times and $k = 2$. Our assumption, stated in terms of intervals; the lengths of the intervals are arbitrary and each horizontal line intersects exactly two intervals. Max SNP-hardness of the problem under consideration (denoted JISP-2) was established by Spieksma [7]. We note that when $k = 2$, Keil's observation [5] can be used to determine whether all the tasks can be scheduled in $O(n)$ time (the problem reduces to 2-SAT). When all the tasks cannot be scheduled using Keil's approach, it is natural to determine the maximum number of tasks that can be scheduled.

We develop two simple and fast randomized approximation algorithms for the JISP-2 problem that have performance ratio strictly greater than 1/2. The second algorithm is derandomized using the method of conditional expectation [3]. To the best of our knowledge the derandomized algorithm is the first deterministic algorithm with performance ratio strictly greater than 1/2. Furthermore, it should be noted that the running times of our algorithms are considerably better than the previous LP based approaches. A naive implementation of the randomized algorithms have running time $O(n^2 \log n)$ and $O(n^2)$. The deterministic algorithm has running time

$O(n^3)$.

2 Randomized Algorithm

In this section we demonstrate a 0.5131057527 factor randomized approximation algorithm for the optimization version of the problem when $k = 2$. Note that our algorithm does not constrain the processing times.

Input is a set $T = \{l_1, r_1, l_2, r_2, \ldots, l_n, r_n\}$ of $2n$ intervals where intervals l_i, r_i lie on horizontal line $y = i$. Denote the set of left-intervals $\{l_1, l_2, \ldots, l_n\}$ using T^l and the corresponding set of right-intervals labelled $r's$ is denoted T^r. Let O be the largest independent set in T.

Let us order the intervals in the increasing order of the right end points. Greedily select a set of intervals J (starting from the left) such that no two intervals intersect a vertical line. Two intervals might intersect the same horizontal line. Note that $|J| \geq |O|$ where O is the optimal solution. Partition the set J into two sets J_1 and J_2, where J_1 is the set of intervals $l_h(r_h)$ in J such that the other interval $r_h(l_h)$ that lies on horizontal line h is not in J, $J_2 = J \setminus J_1$. Intervals in J_1 are independent, whereas intervals in J_2 are not (as each pair lies on a horizontal line). Given J, partition the intervals in the O (the optimal solution) as follows: O_1 is the set of intervals that belong to J, O_2 is the set of intervals that intersect with a pair l_j, r_j in J_2 for some horizontal line j; l_j, r_j are also referred to as the *supports* for $o \in O_2$. $O_3 = O \setminus \{O_1 \cup O_2\}$. Let $|O_3| = (|J_1| + |J_2|)/\beta$, note that $\beta \geq 1$.

J can be thought of as a schedule in which some task(s) may be scheduled twice. One way to obtain a feasible schedule is to choose one of the intervals in J_2 for each task that is scheduled twice. There exist examples where for an arbitrary choice of these intervals, the performance ratio of the algorithm is no better than $1/2$. Let J_2^r be the set of right-intervals from J_2, then $J_1 \cup J_2^r$ is an independent set. The following is the recursive algorithm (randomized) to determine an independent set R.

R = independent(T) {

1. If T is empty then return.
2. Determine J as outlined above.
3. Select elements in J_1 with probability 1/2 each and add them to J'.
4. For each horizontal line j in J_2, add one of l_j or r_j to J' with equal probability.
5. T' is the set of intervals from T that do not intersect (using either horizontal or vertical line) any interval in J'.
6. Determine the largest independent set R' in T' recursively (R'= independent(T')).
7. Return $R' \cup J'$.

}

Finally return the larger of the two independent sets R and $J_1 \cup J_2^r$ as the result. Next we examine the running time. Steps 2–5 can be implemented in $O(n^2)$ after an $O(n \log n)$ preprocessing (sort) step. Every interval in O_2 (subset of the optimal solution) belongs to T' with probability at most $1/2$, i.e., the expected number of intervals from the optimal solution that remain in the subproblem is $|O|/2$. Hence, the total number of steps in the recursion is bounded by $O(\log |O|)$. Therefore, the running time is at most $O(n^2 \log n)$. It might be possible to improve the running time using data structures to maintain deletions in the underlying interval graph. Next, we give an upper bound on β (in terms of α) and a lower bound on the performance ratio α.

Lemma 1 $\beta \leq \frac{2\alpha}{1-\alpha}$

Proof. Each interval $o \in O_2$ has a distinct pair of supports in J_2. Futhermore, none of the supports can be a member of O_1. Hence, $|J_1| + |J_2^r| = |J_1| + |J_2|/2 \geq |O_1| + |O_2|$. The performance ratio α is

$$\geq \frac{|J_1| + |J_2|/2}{|O_1| + |O_2| + |O_3|}$$

$$\geq \frac{|J_1| + |J_2|/2}{|J_1| + |J_2|/2 + |O_3|}$$

$$\geq \frac{|J_1| + |J_2|/2}{|J_1| + |J_2|/2 + (|J_1| + |J_2|)/\beta}$$

$$\geq \min\left\{\frac{\beta}{1+\beta}, \frac{\beta}{\beta+2}\right\} = \frac{\beta}{\beta+2}$$

. □

The next Lemma gives a lower bound on the performance ratio based on the independent set constructed by the recursive algorithm.

Lemma 2 $\alpha \geq 1/2 + \alpha/(\beta 2^\beta)$.

Proof. If O' is the maximum independent set in the subproblem T' then in the recursive step (line 6) we are guaranteed to return an independent set R' of size $\alpha|O'|$. Hence,

$$|R'| + |J'| \geq \frac{|J_1| + |J_2|}{2} + \alpha|O'|.$$

Next, we establish a lower bound on $|O'|$. Consider the bipartite graph whose vertices are the sets J and O_3 respectively. Vertices $j \in J, o \in O_3$ are connected by an edge if and only if the corresponding intervals intersect a vertical line. Let d_o be the degree of vertex $o \in O_3$. The probability that $o \in T'$ is $1/2^{d_o}$. Therefore the expected number of intervals in O_3 that belong to T' is $\sum_{o \in O_3} 1/2^{d_o}$. Each vertex $j \in J$ has degree $d_j \leq 2$, else there is an interval $o \in O_3$ contained in interval j and this violates the fact that j was before o in the ordering (sorted according to the right-end points). Therefore $\sum_{o \in O_3} d_o \leq 2(|J_1| + |J_2|)$. Hence, the minimum number of intervals from O_3 that belong to T' is

given by the following program P:

$$\text{minimize} \sum_{o \in O_3} 1/2^{d_o}$$

$$\text{subject to} \sum_{o \in O_3} d_o \leq 2(|J_1| + |J_2|)$$

The objective function in P is convex and the minimum is achieved when all the variables attain the same value $d_o = 2(|J_1| + |J_2|)/|O_3|$. The minimum value $\geq |O_3|/2^{2(|J_1|+|J_2|)/|O_3|}$. Therefore $|O'| \geq |O_3|/2^{2(|J_1|+|J_2|)/|O_3|}$. Hence,

$$|R'| + |J'| \geq \frac{|J_1| + |J_2|}{2} + \frac{\alpha |O_3|}{2^{2(|J_1|+|J_2|)/|O_3|}}$$

Noting that $|O_3|\beta = |J_1| + |J_2|$, we get

$$|R'| + |J'| \geq \frac{|J_1| + |J_2|}{2} + \frac{\alpha(|J_1| + |J_2|)}{\beta 2^{2\beta}}$$

Therefore the performance ratio $\alpha = (|R'| + |T'|)/(|J_1| + |J_2|)$ is

$$\geq \frac{1}{2} + \frac{\alpha}{\beta 2^{2\beta}}$$

. □

Theorem 1 *The performance ratio of the randomized algorithm is* ≥ 0.5131057527

Proof. Substituting the upper bound for β due to Lemma 1 in Lemma 2, we get

$$\alpha = \frac{1}{2} + \frac{1 - \alpha}{2 \times 2^{\left(\frac{4\alpha}{1-\alpha}\right)}}.$$

Solution for the previous equation (due to Maple) is

$$\frac{LambertW\,(1/4\,\ln(2)) + 4\,\ln(2)}{8\,\ln(2) + LambertW\,(1/4\,\ln(2))} = 0.5131057527$$

□

3 Deterministic Algorithm

In this section we first present a randomized algorithm that is a slight modification of the algorithm presented in the previous section. Next, we derandomize the algorithm using the method of conditional expectation [3]. This derandomization yields the first deterministic algorithm for the problem with a performance ratio strictly better than $1/2$. The algorithm of Chuzhoy et. al. [2] though has a better performance ratio. However, it has a higher running time and is a randomized algorithm.

Once again the algorithm first computes the set J, and then randomly chooses elements (denoted R) from the set J as follows. Elements in J_1 are chosen with probability $1/2$. Next, instead of solving the subproblem recursively (line 6), we compute set $\bar{J}$ (as in Line 2) and return $R \cup \bar{J}_1 \cup \bar{J}_2^r$ as the solution. First we sort the intervals according to their right end points. Step 2–5 take $O(n^2)$ time. Hence, the running time is $O(n^2)$.

Theorem 2 *Performance ratio of the algorithm is* $\geq .5128269905$.

Proof. The analysis is quite similar to the one in Theorem 1. The independent set discovered by the solution has size ($|\bar{J}| \geq |O_3|$)

$$\geq \frac{|J_1| + |J_2|}{2} + \frac{|O_3|}{2 \times 2^{\frac{2(|J_1|+|J_2|)}{O_3}}}$$

Subsituting for $|O_3|$ and dividing by $|J_1| + |J_2|$, we get that the performance ratio α is

$$\geq \frac{1}{2} + \frac{1}{2\beta 2^{2\beta}}$$

Substituting the upper bound for β (from Lemma 1) we get

$$\alpha \geq \frac{1}{2} + \frac{1 - \alpha}{4\alpha 2^{\left(\frac{4\alpha}{1-\alpha}\right)}}$$

Solving for α (using Maple) we get $\alpha = .5128269905$. □

Next, we derandomize the algorithm using the method of conditional expectation implicitly present in [3]. Let T_1 be the set of intervals in $T \setminus \{J \cup O_2\}$. Let $\bar{J}_1 \subseteq T_1$ be the set as computed in Line 2 of the algorithm presented in the previous section. It is important to note that $|O_3| \leq |\bar{J}_1|$. Let the degree (number of intervals in J that vertically intersect o) of each $o \in \bar{J}_1$ be d_o then the probability that $o \in T'$ after the randomized selection (lines 3 and 4) is $\frac{1}{2^{d_o}}$. Given $R \subseteq J$, consider the following function w that assigns weights to the elements of $\bar{J}_1$.

$$w(o) = \begin{cases} 0 & \text{if } o \text{ intersects with some element of } R; \\ 1/2^m & \text{otherwise, where } o \text{ intersects with } m \\ & \text{intervals in } J \setminus R. \end{cases}$$

$w(o)$ is equal to the probability that o will be in the subproblem T' (line 6), given some $R \subset J$. $W = \sum_{o \in \bar{J}_1} w(o)$ when $R = \phi$ is the expected number of intervals from $\bar{J}_1$ that are in the subproblem T'.

Consider the following algorithm: order the elements in J and consider the intervals according to this ordering. We put each element $j \in J$ into R or $S = J \setminus R$ as follows: Either $j \in R$ or $j \in S$. For each of these possibilities compute the weights $w(o)$ for all $o \in \bar{J}_1$. If the sum of the weights is larger for the former case then j belongs to R otherwise j belongs to S. Next we consider the running time. For each $j \in J$ the weights for all $o \in \bar{J}_1$ can be computed in $O(n^2)$

time. Therefore, the total running time is $O(n^3)$.

Theorem 3 *The number of independent intervals in the sub-problem ($\subseteq \bar{J}_1$) after constructing R as above is at least W.*

Proof. Consider the k^{th} stage. Let $w_R(w_S)$ be the sum of weights of elements in $\bar{J}_1$ when the $k+1^{th}$ interval $\in R(S)$. Let w be the sum of weights of the elements in $\bar{J}_1$ at the start of the k^{th} stage. Then, by definition of expectation $w = (w_R + w_S)/2$. According to the algorithm outlined above $j \in R$, if $w_R \geq w_S$ else $j \in S$. Larger of w_R, w_S will be at least w. Note that w is W at the start and never decreases in any stage. Hence, the number of intervals in $\bar{J}_1$ that do not intersect with any interval from R is at least W at the end of the algorithm. □

Note that the size of the independent set constructed by the modified algorithm is

$$\geq (|J_1| + |J_2|)/2 + W/2,$$

using reasoning similar to the one used in Lemma 2, we get,

$$\geq (|J_1| + |J_2|)/2 + 1/2(|\bar{J}_1|)/2^{\frac{2(|J_1|+|J_2|)}{|J_1|}}$$

$$\geq (|J_1| + |J_2|)/2 + 1/2(|O_3|)/2^{\frac{2(|J_1|+|J_2|)}{|O_3|}}.$$

Hence, the analysis in Theorem 2 holds.

To elucidate, consider the example shown in Figure 1. There are 9 tasks labelled $1, 2, 3, A, B, C, D, E, F$. The left and the right intervals corresponding to a task lie on the same horizontal line. Set J is all the intervals labelled $1, 2, 3$ and set $O = \bar{J}_1$ is all the intervals labelled A, B, C, D, E, F. l_i is the left interval for task i and r_i is the right interval for task i. Figure 2 shows the weight for individual elements of O given the assignments to elements in J. In the first stage we consider whether the left or the right interval associated with the task 1 belongs to set R. Note that, $w(O) = 1/2(w(O|l_1) + w(O|r_1)$. In the first stage the algorithm will pick the right interval associated with the first task. Next, we consider the intervals associated with task 2, this is contingent upon $r_1 \in R$. Note that $w(O|\{r_1\}) = 1/2(w(O|\{r_1, l_2\}) + w(O|\{r_1, r_2\}))$. In this second stage we pick r_2 and add it to R. Finally we consider the third task. Once again $w(O|\{r_1, r_2\}) = 1/2(w(O|\{r_1, r_2, l_3\}) + w(O|\{r_1, r_2, r_3\}))$ and we pick r_3.

4 Conclusions

We consider the problem of determining the largest independent set from a set of horizontal intervals, where two intervals are said to be independent if there is no axis-parallel line that intersects them. We study the Max SNP-hard version of the problem where each horizontal line intersects exactly two intervals and give faster approximation algorithms. To the best of our knowledge our deterministic algorithm is the first deterministic algorithm with a performance ratio strictly greater than $1/2$. It appears that the analysis presented here is not tight. An open problem is to improve the analysis.

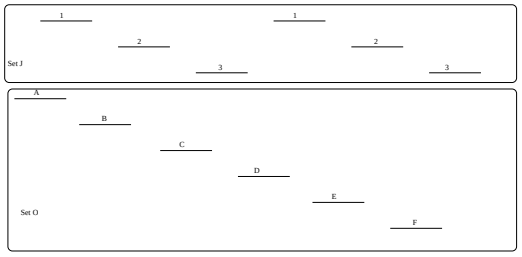

Figure 1: Derandomization Example

	Set R	w(A)	w(B)	w(C)	w(D)	w(E)	w(F)	w
0	$\{\}$	1/2	1/4	1/4	1/4	1/4	1/4	7/4
1	$\{l_1\}$	0	0	1/4	1/2	1/2	1/4	3/2
1	$\{r_1\}$	1	1/2	1/4	0	0	1/4	2
2	$\{r_1, l_2\}$	1	0	0	0	0	1/2	3/2
2	$\{r_1, r_2\}$	1	1	1/2	0	0	0	5/2
3	$\{r_1, r_2, l_3\}$	1	1	0	0	0	1	2
3	$\{r_1, r_3, l_3\}$	1	1	1	0	0	0	3

Figure 2: Weights for the Example

References

[1] M. Adler, A. L. Rosenberg, R. K. Sitaraman and W. Unger. 'Scheduling time-constrained communication in linear network', *In Proc. of* $10th$ *Annual ACM Symposium on Parallel Algorithms and Architectures*, (1998), pp. 269–278.

[2] J. Chuzhoy, R. Ostrovsky, Y. Rabani, 'Approximation Algorithms for the Job Interval Selection Problem and Related Scheduling Problems', $42nd$ *Symposium on Foundations of Computer Science* (2001), pp. 348–356.

[3] P. Erdos and J. L. Selfridge, 'On a Combinatorial Game', *Journal of Combinatorial Theory (B)* 14 (1973), pp. 298–301

[4] T. Erlebach and F. C. R. Spieksma, 'Interval selection: applications, algorithms, and lower bounds', *Journal of Algorithms*, 46(1),(2003) 27–53.

[5] J. M. Keil, 'On the complexity of scheduling tasks with discrete starting times', *Operations Research Letters*, 12 (1992) 293–295.

[6] K. Nakajima and S. L. Hakimi, 'Complexity results for scheduling tasks with discrete starting times', *Journal of Algorithms*, 3 (1982) pp. 344-361.

[7] F. C. R. Spieksma, 'On the approximability of an interval scheduling problem', *Journal of Scheduling*, 2 (1999) pp. 215–227.

[8] F. C. R. Spieksma and Y. Crama, 'The complexity of scheduling short tasks with few starting times', Research Report M92–06, Department of Mathematics, Maastricht University, 1992.

[9] D. K. Y. Yau and S. S. Lam, 'Adaptive rate-controlled scheduling for multimedia applications', *IEEE/ACM Transactions on Networking*, 5(4) (1997) pp. 475-488.