A 1/4 approximation algorithm for a scheduling problem

Problem proposed by Professor J. Mitchell at the CS algorithms seminar on 08/28/2015

January 12, 2016

Problem formulation

Let $E = \{(a_i, d_i)\}_{i=1}^{n}$ be a set of n events ($\{a_i\}$ *arrivals*, $\{d_i\}$ *departures*). Place events at locations in the unit interval so that the minimum distance between events over time is maximized, that is

$$\max \int_0^T d_{\min}(t)\, dt,$$

where $d_{\min}(t)$ is the minimum distance between ongoin events by time t. This situation can be visualized in the cartesian plane, where the unit interval is located on the horizontal axis and the vertical axis represents the timeline, Fig. 1.

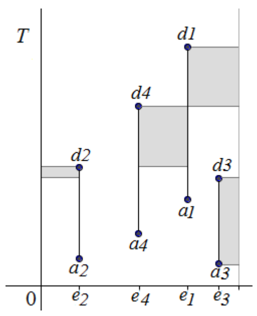

Fig. 1 Graphical representation of event allocation in the unit interval.

To solve this problem we propose an approximation algorithm that consists basically of two steps, a *sweep line stage* and a *breadth first search stage*. In order to achieve a $\frac{1}{4}$ ratio from optimal, job placements will be restricted to dyadic points $\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{3}{8}, \ldots$.

BFS search

The guiding principle here is: "place jobs at their *best available* dyadic point". By availability we mean that at a particular dyadic point there is no job placed there, or that jobs (s_i, d_i) already placed there do not overlap with an incoming job. We will use a binary search tree BST to store jobs at its nodes, because of the bijective correspondence between nodes and dyadic points. By best we mean a dyadic point whose node is as high and to the left as possible in the BST. We will explore the BST in a BFS fashion. As a note, once a job is placed at a point then it remains fixed there. The following pseudo code explains how to proceed

```
BFS_StageAlgorithm
jobs ← $\{(s_i, d_i)\}_{i=1}^{n}$
BST ← new BST( )
for $i$ in $\{1, 2, \ldots, n\}$:
    for node in BST:
        if node is empty or $(s_i, d_i) \cap (s_r, d_r) = \emptyset \; \forall \; (s_r, d_r) \in$ node:
            store $(s_i, d_i)$ in node
    end for
end for
```

Sweep line stage

In the spirit of Bentley-Otmann's algorithm we sweep a line SL bottom-top through the intervals (s_i, d_i). The events $\{s_i\} \cup \{d_i\}$ are stored in an event queue EQ according to earliest occurrence. Now, let us define a *section* as $\delta_k = (e_k, e_{k+1})$ for $e_k, e_{k+1} \in \mathrm{EQ}$ and $|\delta_k| = e_{k+1} - e_k$. Let us define also $d_{\min}(\delta_k) = d_{\min}(t)$ for $t \in \delta_k$ and $\delta(s_i, d_i) = \{\delta_k \mid (s_i, d_i) = \bigcup \delta_k\}$. It should be clear that with this notation

$$\max \int_0^T d_{\min}(t)\, dt = \max \sum_k d_{\min}(\delta_k) \cdot |\delta_k|,$$

where each term $d_{\min}(\delta_k) \cdot |\delta_k|$ can be regarded as the *area contribution* of section δ_k to the objective value. During the sweep line stage we keep track of the number of jobs (s_i, d_i) that intersect with SL in any given section δ_k, we denote this number as $N(\delta_k)$. Also, for $N \in \mathbb{N}$ we define $D(N) = \left\{\frac{i}{2^{\lfloor \log N \rfloor + 1}} \mid i = 1, \ldots, 2^{\lfloor \log N \rfloor + 1} - 1\right\}$. We call $D(N)$ the *best dyadic point set* for N jobs. In the next table we show how this concepts relate

$N(\delta_k)$	$d_{\min}(\delta_k) = \frac{1}{2^{\lfloor \log N(\delta_k) \rfloor + 1}}$	$D(N(\delta_k))$
1	$\frac{1}{2}$	$\left\{\frac{1}{2}\right\}$
2, 3	$\frac{1}{4}$	$\left\{\frac{1}{4}, \frac{2}{4}, \frac{3}{4}\right\}$
4, 5, 6, 7	$\frac{1}{8}$	$\left\{\frac{1}{8}, \frac{2}{8}, \frac{3}{8}, \frac{4}{8}, \frac{5}{8}, \frac{6}{8}, \frac{7}{8}\right\}$
$\vdots$	$\vdots$	$\vdots$

Also, we will construct a boolean matrix whose entries A_{ij} represent if for job (s_i, d_i) there is a section $\delta_k \in \delta(s_i, d_i)$ with $N(\delta_k) = j$. To be more precise

$$A_{ij} = \begin{cases} 1 & \text{if exists } \delta_k \in \delta(s_i, d_i) \text{ s.t. } N(\delta_k) = j \\ 0 & \text{otherwise} \end{cases}$$

Here the number of columns $m = \max_k N(\delta_k)$. This matrix will help us decide in which order jobs must be processed by **BFS_StageAlgorithm**, the justification for this $w$ill be discussed later. To this preprocessing stage we will refer as **Sweep_Algorithm**.

Final stage

Once the sweep line stage is finished, we process jobs with **BFS_StageAlgorithm**, starting with those jobs such that $N(\delta_k) = 1$ for some $\delta_k \in \delta(s_i, d_i)$, i.e. $j = 1$; then continuing with those *unselected* jobs (not yet placed) such that $N(\delta_k) = 2$ for some $\delta_k \in (s_i, d_i)$, i.e. $j = 2$; and so forth. In this sense, the matrix will be explored columnwise left-to-right to select an order to process jobs with **BFS_StageAlgorithm**.

```
Final_StageAlgorithm
Q ← new queue( );
for $j = 1, ..., m$:
    for $i = 1, ..., n$:
        if $A_{ij} = 1$ and $(s_i, d_i)$ is unselected:
            push $(s_i, d_i)$ into Q
    end for;
end for;
Process Q with BFS_StageAlgorithm
```

Example

Before we continue with the proof, we give a small example to illustrate these ideas. Consider the following instance $\{job_1(1,8), job_2(0,6), job_3(2,4), job_4(3,5)\}$. After applying **Sweep_Algorithm** we obtain the table

Event	EQ	δ_k	$N(\delta_k)$	$D(N(\delta_k))$	Matrix's rows status
$e_1 = s_2$	$s_2 s_1 s_3 s_4 d_3 d_4 d_2 d_1$	δ_1	1	$\{\frac{1}{2}\}$	$A_2[1]$
$e_2 = s_1$	$s_1 s_3 s_4 d_3 d_4 d_2 d_1$	δ_2	2	$\{\frac{1}{4}, \frac{2}{4}, \frac{3}{4}\}$	$A_2[1,1],\ A_1[0,1]$
$e_3 = s_3$	$s_3 s_4 d_3 d_4 d_2 d_1$	δ_3	3	$\{\frac{1}{4}, \frac{2}{4}, \frac{3}{4}\}$	$A_2[1,1,1],\ A_1[0,1,1],\ A_3[0,0,1]$
$e_4 = s_4$	$s_4 d_3 d_4 d_2 d_1$	δ_4	4	$\{\frac{1}{8}, \ldots, \frac{7}{8}\}$	$A_2[1,1,1,1],\ A_1[0,1,1,1],\ A_3[0,0,1,1],\ A_4[0,0,0,1]$
$e_4 = d_3$	$d_3 d_4 d_2 d_1$	δ_5	3	$\{\frac{1}{4}, \frac{2}{4}, \frac{3}{4}\}$	$A_2[1,1,1,1],\ A_1[0,1,1,1],\ A_4[0,0,1,1]$
$e_5 = d_4$	$d_4 d_2 d_1$	δ_6	2	$\{\frac{1}{4}, \frac{2}{4}, \frac{3}{4}\}$	$A_2[1,1,1,1],\ A_1[0,1,1,1]$
$e_6 = d_2$	$d_2 d_1$	δ_7	1	$\{\frac{1}{2}\}$	$A_1[1,1,1,1]$
$e_7 = d_1$	d_1		0		

and the boolean matrix

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix},$$

then by applying **BFS_StageAlgorithm** we obtain a solution with an objective value of $\frac{9}{4}$, Fig. 2.

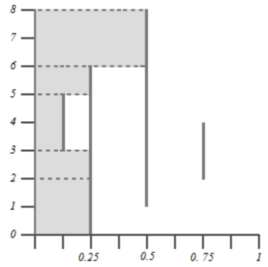

Fig. 2 The filled zones represent the area contribution.

1/4 approximation

If jobs could be arranged *independently* from each other at each section δ_k, then the best points for jobs that span δ_k would be $\frac{1}{1+N(\delta_k)}, \ldots, \frac{N(\delta_k)}{1+N(\delta_k)}$, in such case $d_{\min}(t) = \frac{1}{1+N(\delta_k)}$ and the following inequalities hold true

$$\frac{1}{2(1+N(\delta_k))} \leq \frac{1}{2^{\lfloor \log N(\delta_k) \rfloor + 1}} \leq \frac{1}{1+N(\delta_k)}$$

Let OPT be an optimal solution and let's denote by OPT_{δ_k} the area contribution of OPT to section δ_k. Now, if jobs that span δ_k are placed at points in $D(N(\delta_k))$ then it follows

$$\frac{OPT_{\delta_k}}{2} \leq \frac{|\delta_k|}{2(1+N(\delta_k))} \leq \frac{|\delta_k|}{2^{\lfloor \log N(\delta_k) \rfloor + 1}} \leq \frac{|\delta_k|}{1+N(\delta_k)},$$

so if we add up the previous inequalities over all δ_k we could claim that an approximate solution no worse than $\frac{1}{2}OPT$ has been obtained. However, placing jobs at their best dyadic point set per section (to this we refer as *dyadic optimality*) is not always achievable, this is shown in Fig. 3

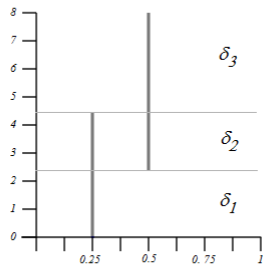

Fig. 3 Jobs $(0, 4.5)$ and $(2.5, 8)$ overlap at δ_2 so they can't be both placed at $\frac{1}{2}$ to achieve dyadic optimality for sections δ_1 and δ_3.

In the previous example, section δ_3 achieves dyadic optimality and δ_1 achieves *half* of its dyadic optimality. We will prove that our algorithm yields a solution in which some sections will achieve dyadic optimality, and some sections will achieve half of its dyadic optimality in the worst case. This is the core idea to prove our $\frac{1}{4}$ factor approximation ratio, that we will state as **Lemma 01**.

Before proving our lemma, let us introduce the following notation that will be useful in our proof

$$\{(s_i, d_i)\}_j = \left\{ (s_i, d_i) \mid \min_{\delta_k \in \delta(s_i, d_i)} N(\delta_k) = j \right\},$$

The sets $\{(s_i, d_i)\}_j$ represent clusters of jobs according to the following: all jobs in $\{(s_i, d_i)\}_j$ have their first non-zero entry at the *jth* column of $[A_{ij}]$, and clearly for some values of j the respective set could be empty, nevertheless $\{(s_i, d_i)\}_{i=1}^{n} = \cup \{(s_i, d_i)\}_j$.

Lemma 01 Let $\{(s_i, d_i)\}_{i=1}^{n}$ be an instance of our problem. Compute the $n \times m$ matrix $[A_{ij}]$ with **SweepAlgorithm** and the queue Q with **Final_StageAlgorithm**, then, for $j = 1, 2, \ldots, m$ and for each $(s_i, d_i) \in \{(s_i, d_i)\}_j$ when processed with **BFS_StageAlgorithm**

1. Job (s_i, d_i) will be placed at some dyadic point in $D(j)$ if there is an available point (no overlapping with other jobs previously placed), else
2. Job (s_i, d_i) will be placed at some dyadic point in $D(2j)$.

We proceed by induction on j.

Proof

i) If $\{(s_i, d_i)\}_1 = \emptyset$ our assertion for $j = 1$ is vacuously true, so suppose that $\{(s_i, d_i)\}_1 \neq \emptyset$. Now, let $(s_{i_1}, d_{i_1}), (s_{i_2}, d_{i_2}) \in \{(s_i, d_i)\}_1$ such that both jobs overlap. Without loss of generality suppose that $s_{i_1} < s_{i_2}$, then it is necessary that $d_{i_1} < d_{i_2}$, and the overlapping section is (s_{i_2}, d_{i_1}). It should be clear that no other job in $\{(s_i, d_i)\}_1$ can overlap at (s_{i_2}, d_{i1}) without covering (s_{i_1}, s_{i_2}) or (d_{i_1}, d_{i_2}) (if this happens then one job is not in $\{(s_i, d_i)\}_1$), so the maximum overlapping is $2^{\lfloor \log(1) \rfloor + 1} = 2$. Now, by the way **BFS_StageAlgorithm** places jobs, it should be clear that $\frac{1}{2}$ is the choice when availability holds, and $\frac{1}{4}$ is the choice when there is overlapping, so $D(2) = \left\{\frac{1}{2}, \frac{1}{4}, \frac{3}{4}\right\}$ will be sufficient to place all jobs $\{(s_i, d_i)\}_1$.

ii) Suppose now that our assertion is valid for $m - 1$. Let's prove it for m.

iii) If $\{(s_i, d_i)\}_m = \emptyset$ our assertion for $j = m$ is vacuously true, so suppose that $\{(s_i, d_i)\}_m \neq \emptyset$. ***This is the tricky part left to be proven, nevertheless I feel that the lemma is true.***

The direct implication of lemma 01 is that at each section we can achieve at least half dyadic optimality, so the following inequalities hold true for every section δ_k

$$\frac{OPT_{\delta_k}}{4} \leq \frac{|\delta_k|}{4(1 + N(\delta_k))} \leq \frac{|\delta_k|}{2^{\lfloor N(\delta_k) \rfloor + 2}},$$

so if we add this inequalities over all δ_k our approximate solution is no worse than $\frac{1}{4}$ of OPT.

Practical improvement

An improvement, not to the approximation's ratio but to the objective's value, can be made if at ***Final_ StageAlgorithm*** we select jobs at each column of the matrix $[A_{ij}]$ with some priority. In order to achieve this, we must use a modified version of $[A_{ij}]$

$$\tilde{A}_{ij} = \sum_{\substack{\delta_k \in \delta(s_i, d_i), \\ N(\delta_k) = j}} |\delta_k|$$

In this modified version, those entries that had a zero in $[A_{ij}]$ will still have a zero in $\left[\tilde{A}_{ij}\right]$, but instead of having one, an entry $\tilde{A}_{ij}$ will have the total length of those sections that satisfy $N(\delta_k) = j$, and that are spanned by the *ith* job. Then, when jobs are being added to the queue Q at ***Final_ StageAlgorithm*** during the jth column iteration, they will be selected according to the value at its respective jth entry (if this is non-zero and if job has not yet been placed, of course), the greater the value, the greatest the priority. By doing so, when **BFS_StageAlgorithm** processes jobs, those jobs that span larger areas will be given priority to be placed at better dyadic points, thus allowing the possibility of achieving a better objective value.

Randomized approach for online version

For the online version of the problem, where the (s_i, t_i) are unknown in advance, a randomized approach could be taken as follows: "for an incoming job, choose the best available dyadic point with a uniform distribution". An optimistic conjecture would be to prove that this yields and expected constant time approximation. If this approach is succesful, it could be extended naturally to two or more dimensions.

Two dimensional version.

This ideas can be extended to the unit square, if we apply the concepts on both of its defining unit intervals. Now jobs would be located at points in the plane (x_i, y_i) where each coordinate is a dyadic point. Our search tree would have a branching factor of 4 instead of two and our objective function would be

$$\max \int_0^T a_{\min}(t)\, dt,$$

where $a_{\min}$ stands for the minimum area of the square that can be placed between points. Our previous notion of area contribution would now be replaced with that of volume contribution, as we would interpret time as the $z - axis$. Left is to prove the constant factor approximation.