Common-Deadline Lazy Bureaucrat Scheduling Problems

Behdad Esfahbod, Mohammad Ghodsi, Ali Sharifi

{behdad,ghodsi}@sharif.edu, ali@bamdad.org

Computer Engineering Department
Sharif University of Technology
Tehran, Iran

Introduction

- Scheduling problems: A number of jobs to be performed by some employees.

- Common studies: To be as efficient as possible. The *employer*'s point of view.

- Lazy bureaucrat: To be as inefficient as possible. The *lazy employee*'s point of view. These are called *Lazy Beauracrat Scheduling Problems* (LBSP).

- Some restrictions are needed to get out of degenerate cases.

- Similar considerations: Shortest vs longest path problems.

Problem Definition

- A set J of jobs $j_1, \ldots, j_n$, processing times (job lengths) $t_1, \ldots, t_n$, arrival times $a_1, \ldots, a_n$, deadlines $d_1, \ldots, d_n$.

- Jobs should be processed completely, if ever. We consider the case with no preemption.

- Hard deadlines. Processed jobs should be done so not before their arrival neither after their deadline. We call the interval $[a_i, d_i]$ the *window* for job i.

- Offline scheduling. We now the arrival times and deadlines beforehand.

- All the numbers are non-negative integers. Assume WLOG that there is a job arriving at time 0. Let $D = max(d_i)$.

More Definitions

- **Executable Job:** Job j_i is *executable* at time t, iff it is not processed yet, and $a_i \leq t \leq d_i - t_i$.

- **Greedy Requirement:** At any time, the bureaucrat should work on an executable job, if there is any.

- The goal is to be as inefficient as possible. This is captured by any of the following objective functions that is to minimize.

Objective Functions

1. `[min-time-spent]`: Minimize the total amount of time spent working.

2. `[min-weighted-sum]`: Minimize the weighted sum of completed jobs.

3. `[min-makespan]`: Minimize the makespan, the maximum completion time of the jobs.

4. `[min-number-of-jobs]`: Minimize the total number of completed jobs.

Objective functions 1 and 4 are special cases of 2.
If all jobs have the same arrival time, the objective functions 1 and 3 are equivalent.

Some Previous Results

Non-preemptive and multi-employee cases have also been studied. They are not listed here.

- LBSP is strongly NP-hard under all objective functions and is not approximable to within any fixed factor.

- LBSP with the same arrival times for all jobs, is weakly NP-hard, and can be solved by a pseudo-polynomial dynamic programming algorithm.

- LBSP with all the jobs having unit lengths, can be solved in polynomial time by the Latest Deadline First (LDF) scheduling policy.

Some Previous Results (continued)

- Assuming for each job i, $d_i - a_i < 2t_i$, LBSP can be solved in $O(nD \max(n, D))$ time.

- Even with a bound on δ (the ratio of the longest job to the shortest job), LBSP is strongly NP-hard. It cannot be approximated to within a factor of $\delta - \epsilon$, for any $\epsilon > 0$, unless $P = NP$.

- Given bounds on R (the maximum ratio of window length to job length) and δ, LBSP can be solved in $O(Dn^{4R \lg \delta})$.

- Assuming $d_i - a_i < 2t_i$ for each job i ($R \leq 2$), LBSP can be solved in $O(nD)$ time.

Our Problem — Notation

- **Common-Deadline LBSP (**CD-LBSP**):** When deadlines for all jobs are the same (D).

- CD-LBSP`[objective-function]`: When considered with one of defined objective functions.

- CD-LBSP`[*]`: All/any of the objective functions.

Results

- CD-LBSP`[*]` is still NP-hard.

- CD-LBSP`[min-number-of-jobs]` is not approximable to within any fixed factor.

- There is a tight 2-approximation algorithm for CD-LBSP`[min-makespan]`.

- CD-LBSP`[*]` is *weakly* NP-hard: There exists a pseudo-polynomial time dynamic programming algorithm.

NP-Hardness

Theorem 1 CD-LBSP[*] *is NP-hard.*

Proof

- Reduce the Subset Sum problem to this problem,
- Given: $S = \{x_1, \ldots, x_n\}$ of n positive integers, where $\Sigma_{i=1}^{n} x_i = s$, and integer b $(0 < b < s)$,
- Is there any $T \subset S$, satisfying $\Sigma_{x \in T}\, x = b$?
- WLOG, we assume that $b \leq \lfloor \frac{s}{2} \rfloor$ and $x_i < b$ for all i.
- Construct an instance of CD-LBSP containing $n + 1$ jobs.

NP-Hardness (continued)

- All deadlines $D = 2s$,
- $\forall x_i \in S$, define job j_i with $a_i = 0$, and $t_i = 2x_i$,
- Define j_{n+1} with $a_{n+1} = 2b$ and $t_{n+1} = 2s - 2b - 1$.
- The employee can finish his work by time $2s$ or $2s - 1$,
- He can finish by $2s - 1$ iff he finds a solution for the Subset Sum problem.

$\square$

Approximablity

Theorem 2 CD-LBSP`[min-number-of-jobs]` *is not approximable to within any fixed factor* $\Delta > 1$*, unless* $P = NP$*.*

Proof

- Reduce Subset Sum problem again! This time to reach contradiction.
- Assume that there is an approximation algorithm with a fixed factor Δ.
- Let $m = \lceil \Delta \rceil$ and $D = b + m(n+2)s$ (a huge number).

Approximablity (continued)

- Construct an instance of CD-LBSP`[min-number-of-jobs]` containing the following jobs, all with deadline D:
  - $\forall x_i \in S$, define an *element job* j_i having $a_i = 0$, and $t_i = x_i$,
  - Define one *long job*, j_{n+1} with $a_{n+1} = b$ and $t_{n+1} = D - b$.
  - And define $m(n+2) - 1$ *extra jobs* (lots of them), all having arrival times b, and processing times s.

- The bureaucrat wants to do as few jobs as possible: He should process the long job, to avoid too many extra jobs,

- So he can process as few as $n + 1$ jobs iff he finds a solution for the Subset Sum problem.

Approximablity (continued)

- The hypothetical Δ-approximation would produce at most $m(n+1)$ jobs.
- There are $m(n+2)-1$ extra jobs, that all can be processed if the long job is not.
- So the approximation algorithm is forced to produce the optimal solution $\Rightarrow$ solution for the Subset Sum problem $\Rightarrow P = NP$.

$\square$

Corollary 1 CD-LBSP`[min-weighted-sum]` *is not approximable to within any fixed factor* $\Delta > 1$, *unless* $P = NP$.

Approximation

Theorem 3 *The Shortest Job First (SJF) scheduling policy is a 2-approximation algorithm for* CD-LBSP`[min-makespan]` *and this bound is tight.*

Proof

- Let σ_{OPT} be an optimal solution and σ be the schedule which the SJF policy has generated, and *OPT* and *SJF* be their makespans respectively,

- We show that $SJF - OPT < OPT$,

- WLOG suppose that $j_1, \ldots, j_k$ are the jobs processed in σ in that order,

- Let $j_q \in \sigma$ be the job satisfying $start_q(\sigma) < OPT \leq finish_q(\sigma)$,

Approximation (continued)

- We know that $a_i < OPT$ for all jobs,

- SJF policy forces that $t_{q+1} \leq t_{q+2} \leq \ldots \leq t_k$,

- Greedy requirement forces that $j_i \in \sigma_{OPT}$ for all $q + 1 \leq i \leq k$

- If $j_q \in \sigma_{OPT}$ then $SJF - OPT < \Sigma_{i=q}^{k} t_i \leq OPT$, and we are done,

- Otherwise, there exists some job in σ_{OPT} that is arrived before j_q, and not shorter than j_q, which is not processed in σ, call it j_p,

- So we have

$$SJF - OPT < t_q + \Sigma_{i=q+1}^{k} t_i \leq t_p + \Sigma_{i=q+1}^{k} t_i \leq OPT.$$

Tightness

- Given n and $0 < \epsilon < 1$, we construct an instance of CD-LBSP with n jobs, which SJF does no better than $2 - \epsilon$,
- All jobs have $a_i = 0$ and $d_i = D$, where $D = n - 3 + 2L - 1 = n + 2L - 4$ with $L = 2n/\epsilon$:
- $t_1 = L - 1$, $t_2 = L$, $t_3 = L + 1$, $t_i = 1$ $(4 \leq i \leq n)$,
- OPT would process all unit jobs and then j_3, having makespan $OPT = n - 3 + L + 1 = n + L - 2$,
- SJF would process $j_4, \ldots, j_n$, j_1, and j_2 having makespan $SJF = n - 3 + L - 1 + L = n + 2L - 4$.
- With some magical math, we would have $\frac{SJF}{OPT} > 2 - \epsilon$. $\square$

Pseudo-Polynomial Time Algorithms

- Assume that the jobs are numbered in order of their arrival times,
- Let $T_i = \{j_i, \ldots, j_n\}$, $T_{i,k} = \{j_i, \ldots, j_k\}$,
- We can assume that in the optimal schedule, the consecutive jobs are performed in order of their arrival times,

Lemma 1 *For a given (T_i, α, U), we can find an optimal schedule without any gaps from some jobs in T_i, and up to time α, so that all jobs in U appear in the schedule, if any such schedule exists, in time $O(n\alpha)$.*

Proof It can be solved much like the binary knapsack problem. $\square$

Pseudo-Polynomial Time Algorithms (continued)

Theorem 4 CD-LBSP[*] *is weakly NP-hard.*

Proof

- Let P_i be the subproblem of scheduling the jobs in T_i. P_1 is the original problem,

- Let α be the first *rest time* in an optimal schedule σ for P_i,

- So the schedule can be broken into two independent subschedules, one before α, and one after α,

- For α to be a rest time, there are some jobs forced to be in the first schedule,

Pseudo-Polynomial Time Algorithms (continued)

- The first subschedule can be found by applying the lemma from previous page,
- The second one is the optimal solution to subproblem P_k, where j_k is the first job arriving after α,
- All we need to do is to search for α.
- This all takes time $O(n^2 D^2)$ to solve P_1.
- So CD-LBSP[*] is weakly NP-hard.

$\square$

Conclusion

We studied a new class of the Lazy Bureaucrat Scheduling Problems (LBSP), called common-deadline LBSP, where the deadlines of all jobs are the same. We proved that this problem is still NP-hard under all four pre-defined objective functions. We also showed that this problem is not approximable to within any fixed factor in cases of `[min-weighted-sum]` and `[min-number-of-jobs]` objective functions. The problem is shown to have a tight 2-approximation algorithm under `[min-makespan]`. But, it is still open whether it is approximable under `[min-time-spent]`. In the rest of the paper, we presented pseudo-polynomial time dynamic programming algorithms for this problem under all objective functions. Further work on this problem is underway.

Acknowledgements The authors would like to thank the anonymous referees for their useful comments.

References

Arkin, E. M., Bender, M. A., Mitchell, J. S. B., Skiena, S. S.: *The lazy bureaucrat scheduling problem*. Workshop on Algorithms and Data Structures (WADS'99), LNCS 1663, pp. 122–133, Springer-Verlag, 1999.

Gary, M. R., Johnson D. S.: *Computers and intractability, a guide to the theory of NP-completeness*. W. H. Freeman and Company, New York, 1979.

Farzan, A., Ghodsi, M.: *New results for lazy bureaucrat scheduling problem*. 7th CSI Computer Conference (CSICC'2002), Iran Telecommunication Research Center, March 3–5, 2002, pp. 66–71.

Hepner, C., Stein, C.: *Minimizing makespan for the lazy bureaucrat problem*. Scandinavian Workshop on Algorithm Theory (SWAT'2002), LNCS 2368, pp. 40–50, Springer-Verlag, 2002.