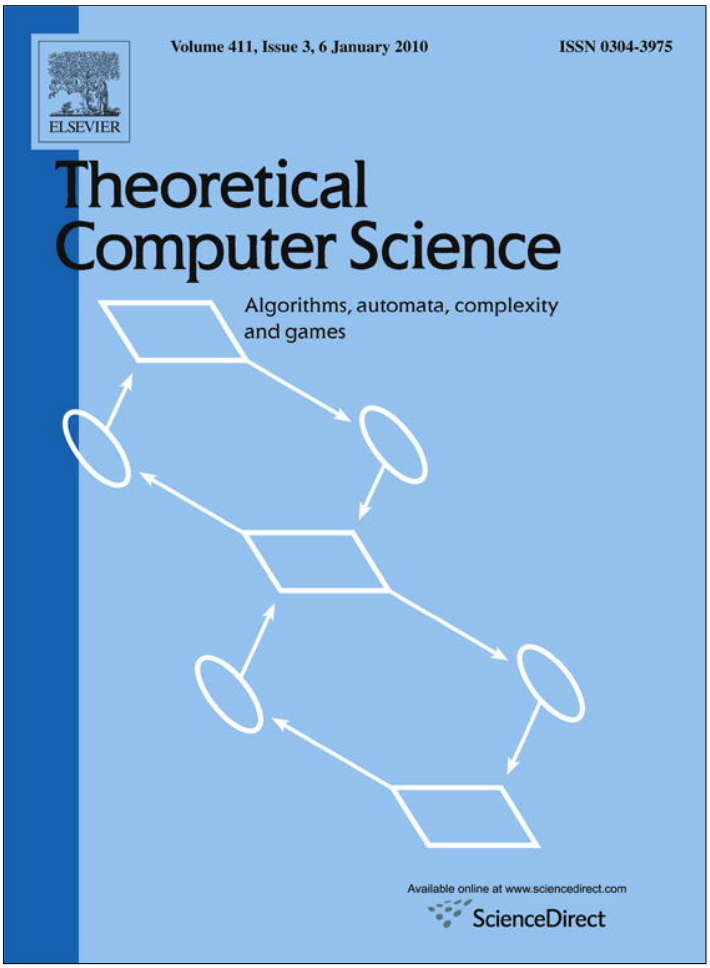

Theoretical Computer Science 411 (2010) 672–676

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Energy optimal schedules for jobs with multiple active intervals

Wanyong Tian [a,b,c], Minming Li [b,*], Enhong Chen [a]

[a] School of Computer Science, University of Science and Technology of China, China
[b] Department of Computer Science, City University of Hong Kong, China
[c] USTC-CityU Joint Research Institute, China

ARTICLE INFO

Article history:
Received 27 May 2009
Received in revised form 3 November 2009
Accepted 7 November 2009
Communicated by Ding-Zhu Du

Keywords:
Dynamic voltage scaling
Multiple interval jobs
Min-energy schedule

ABSTRACT

In this paper, we study the scheduling problem of jobs with multiple active intervals. Each job in the problem instance has n $(n \geqslant 1)$ disjoint active time intervals where it can be executed and a workload characterized by the required number of CPU cycles. Previously, people studied multiple interval job scheduling problem where each job must be assigned enough CPU cycles in one of its active intervals. We study a different practical version where the partial work done by the end of an interval remains valid and each job is considered finished if total CPU cycles assigned to it in all its active intervals reach the requirement. The goal is to find a feasible schedule that minimizes energy consumption. By adapting the algorithm for single interval jobs proposed in Yao, Demers and Shenker (1995) [1], one can still obtain an optimal schedule. However, the two phases in that algorithm (critical interval finding and scheduling the critical interval) can no longer be carried out directly. We present polynomial time algorithms to solve the two phases for jobs with multiple active intervals and therefore can still compute the optimal schedule in polynomial time.

1. Introduction

The energy efficiency problem in processor scheduling has become increasingly important these years, partly because portable electronic devices which are mainly powered by batteries are widely in use nowadays. Among various techniques aiming to provide possibilities for energy efficient schedules, DVS (Dynamic Voltage Scaling) technique has been adopted by major chip manufacturers like Intel, IBM and AMD. The microprocessors which support DVS allow the speed of the processors to be adjusted in runtime so that the goal of energy efficiency can be achieved if jobs are suitable scheduled.

A theoretical study of min-energy DVS scheduling was initiated by Yao, Demers and Shenker [1]. They formulated the optimization problem and gave an $O(n^3)$ algorithm YDS for computing the optimal schedule, which is later improved to $O(n^2 \log n)$ in [2]. In their formulation, each job J_i has an arrival time a_i, a workload R_i (which means the number of CPU cycles this job needs in order to finish), and a deadline b_i by which the job must be finished. We call these types of jobs single interval jobs. If job J_i is run at speed s, then it needs R_i/s time to finish. They also assumed a speed to power function $P(s) = s^\alpha$, where $\alpha \geq 2$ is a system constant (usually $\alpha = 3$ according to the cubic-root rule). Their model allows the processor speed to be set at any real value and any feasible schedule must guarantee that each job J_i is assigned enough CPU cycles within its interval $[a_i, b_i]$.

The seminal paper [1] mentioned above is about the scheduling problems in which each job has a single time interval. There are also some works on the scheduling problem for jobs with multiple time intervals in a processor with constant speed. In their multiple interval setting, a job has n $(n \geqslant 1)$ disjoint time intervals. Such a job could be scheduled in any

* Corresponding author. Tel.: +852 27889538; fax: +852 27888614.
E-mail addresses: frog@mail.ustc.edu.cn (W. Tian), minmli@cs.cityu.edu.hk (M. Li), cheneh@ustc.edu.cn (E. Chen).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.11.002

of its intervals. If the job is incomplete by the end of one of its intervals, the partial work done is lost. When it is scheduled in a later interval, it has to restart from the very beginning. A job fails when it is still incomplete by the end of the last interval. [3] investigated multiple interval model for unit length jobs and showed that the general problem is NP-complete, but for a special case one can determine in polynomial time whether a feasible schedule exists. [4] proved the NP-hardness of the problem of scheduling real-time jobs that have multiple feasible intervals. They presented an exponential optimal algorithm, and gave some polynomial time heuristics. [5] and [6] considered how to find a schedule in which the number of jobs completed in one of their feasible intervals is maximized. They showed that the maximization problem is NP-hard for both non-preemptive and preemptive jobs, and presented two approximation algorithms for them respectively.

Practically, we observe that there exists another kind of multiple interval jobs as follows: A job with a required amount of workload can be executed in certain discontinuous multiple time intervals (we call these intervals active intervals for this job). The job is regarded finished when the sum of workload done in all active intervals achieves the requirement. Thus, the consideration of energy efficient schedules for these types of jobs naturally arises. In this paper, we investigate how to compute a min-energy schedule for jobs with multiple active intervals. Different from [3–6], in our problem, the partial work is not lost by the end of one active interval and the processor we investigate can adjust its speed when executing jobs. We present a polynomial time offline schedule to solve the problem as follows. First we find the critical interval set by adopting a similar method proposed in [1] for single interval jobs, and then schedule the jobs whose intervals are totally included in the critical interval set. For the first step, original methods for single interval jobs [1] needs exponential time and therefore we find another way to calculate the results in polynomial time; for the second step, we observe that EDF scheduling policy, which can guarantee the feasibility for the single interval jobs, cannot guarantee a feasible schedule for the multiple interval jobs, therefore, we use Linear Programming to calculate a feasible schedule in polynomial time. More works investigating different aspects of DVS can be found in [7–14].

The rest of this paper is organized as follows. Section 2 describes the model and gives some definitions. Section 3 gives a polynomial time offline method for the min-energy scheduling problem of multiple interval jobs. Finally, we conclude the paper in Section 4.

2. Preliminaries

We first describe the min-energy scheduling model. Since our model is similar to that described in [1], we adopt the similar notation for consistency and modify slightly when necessary.

Let J be a set of jobs to be executed during time interval [0, 1]. Each job $J_k \in J$ is characterized by parameters below.

- T_k: a set of time intervals in which J_k could be executed. We represent T_k as $T_k = \bigcup_{1\leqslant i\leqslant m_k}[t_{k,i}, t'_{k,i}]$ $(0 \leq t_{k,1} < t'_{k,1} < \cdots < t_{k,m_k} < t'_{k,m_k} \leq 1)$,
- R_k: required number of CPU cycles.

If job J_k is run at constant speed s, then it needs $\frac{R_k}{s}$ time to finish its workload. A schedule S is a pair of functions $(s(t), job(t))$ defined over [0, 1] where $s(t) \geqslant 0$ is the processor speed at time t and $job(t)$ defines the job being executed at time t (or idle if $s(t) = 0$). Both *s(t)* and *job(t)* are piecewise constant with finitely many discontinuities.

A feasible schedule for an instance J is a schedule S that satisfies $\sum_{i=1}^{m_k}\int_{t_{k,i}}^{t'_{k,i}} s(t)\delta(job(t), J_k)\mathrm{d}t = R_k$ for all $J_k \in J$.

Here, $\delta(x, y)$ is 1 if $x = y$ and 0 otherwise, and m_k denotes the number of intervals in T_k. In other words, S must give each job J_k the required number of cycles in its time intervals T_k (with perhaps intermittent execution). The total energy consumed by a schedule S is $E(S) = \int_0^1 P(s(t))dt$, where power function $P(s)$ is usually assumed to be $P(s) = s^\alpha$.

Now we present the formalized definition of our scheduling problem below:

Given a job set $J = \{J_1, J_2, \ldots, J_n\}$, in which each job $J_k (1 \leqslant k \leqslant n)$ requires R_k CPU cycles , and could be executed in a given finite set of time intervals $T_k = \bigcup_{1\leqslant j\leqslant m_k}[t_{k,j}, t'_{k,j}]$ $(t_{k,1} < t'_{k,1} < \cdots < t_{k,m_k} < t'_{k,m_k}, m_k \geqslant 1)$, the objective is to find a min-energy schedule $S = (s(t), job(t))$ where each job J_k $(1 \leqslant k \leqslant n)$ completes workload R_k within T_k.

We should compute both the optimal speed function *s(t)* and the related function *job(t)* which decides the job to be executed at time t.

3. The offline energy optimal algorithm

We first give two definitions below.

Definition 1. Define the intensity of a set of intervals T to be $g(T) = \frac{\sum R_i}{|T|}$, where the summation is taken over all jobs J_i with $T_i \subseteq T$.

Definition 2. Let T^* be a set of intervals that maximizes $g(T)$. We call T^* the critical interval set for J, and call the set of jobs $J_{T^*} = \{J_i | T_i \subseteq T^*\}$ the critical group for J.

The algorithm we propose is quite similar to the algorithm proposed in [1] for single interval jobs. We proceed by identifying the *critical interval set* for J, scheduling jobs whose intervals are totally included in the *critical interval set*, then constructing a subproblem for the remaining jobs and solving it recursively. However, finding the critical interval set and

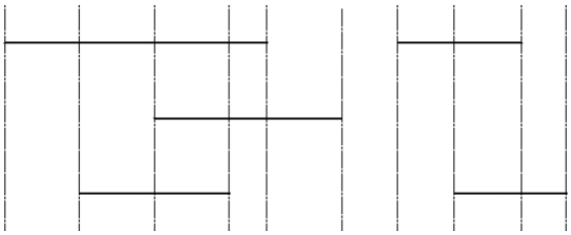

Fig. 1. A 3-job set with 5 intervals and 8 pieces.

scheduling jobs within the critical interval set are no longer straightforward. Firstly, finding the critical interval set by enumerating all the possible T will require exponential time because of the existence of jobs with multiple active intervals. To overcome this, we find that zero-one fractional programming given in [15] can be specialized to solve our problem in polynomial time. Secondly, even if the critical interval set is found, we observe that the commonly used scheduling policy EDF could no longer guarantee a feasible schedule. Therefore, we propose a method based on linear programming to generate a feasible schedule in the critical interval set. The complete algorithm is given in Algorithm 1.

Algorithm 1 *Optimal Schedule for Multiple interval Jobs*

Input: a job set J
Output: Optimal Schedule $S = (s(t), job(t))$

```
while J ≠ ∅ do
  Find the critical interval set T* using the method proposed in Section 3.1.
  Schedule the critical group J_{T*} using a linear programming method proposed in Section 3.2.
  J ← J − J_{T*}
  for all J_k ∈ J do
    for i=1 to m_k do
      a_{k,i} = a_{k,i} − |T* ∩ [0, a_{k,i}]|.
      b_{k,i} = b_{k,i} − |T* ∩ [0, b_{k,i}]|.
    end for
  end for
end while
```

3.1. Finding the critical interval set

We first process our model by dividing the time intervals into *pieces*, and then prove that finding a critical interval is in fact a special case of *zero-one fractional programming* problem proposed in [15].

Given a job set $J = \{J_1, J_2, \ldots, J_n\}$. Let m_i be the number of time intervals of J_i, and there are altogether $\sum_{i=1}^{n} m_i$ original intervals in the input. Each interval has two endpoints. Viewing each endpoint of the intervals as a break point, we can divide the time span [0, 1] into m ($m \leqslant 2 \cdot \sum_{i=1}^{n} m_i - 1$) *pieces*. An instance consisting of 3 jobs with 5 intervals and 8 *pieces* is shown in Fig. 1. Notice that the piece of time during which no jobs can be executed is not counted as a piece. We say a job J_i is covered by a set of pieces if T_i is contained in the union of the set of pieces. We define the density of a set of pieces as the total workload of the jobs covered by this set of pieces divided by the total length of the set of pieces. Then finding a critical interval set can be interpreted as finding a set of pieces (intervals) with maximum density.

Now we describe the *zero-one fractional programming* problem proposed in [15].

Suppose $f(x) \geqslant 0$ and $g(x) > 0$ are given polynomials defined for all $x = (x_1, x_2, \ldots, x_n)$ in $L = \{0, 1\}^n - \{0\}^n$.

$$f(x) = \sum_{P \in A} a_P \prod_{i \in P} x_i + \sum_{i=1}^{n} a_i x_i$$

$$g(x) = \sum_{Q \in B} b_Q \prod_{i \in Q} x_i + \sum_{i=1}^{n} b_i x_i$$

The sets A and B are given collections of non-empty non-singleton subsets of $\{1, \ldots, n\}$, $a_P \geqslant 0$ for each $P \in A$, and $b_Q \leqslant 0$ for each $Q \in B$. Since $f(x) \geqslant 0$ and $g(x) > 0$ for all $x \in L$, we have $a_i \geqslant 0$ and $b_i > 0$ for each $i \in \{1, \ldots, n\}$. The objective is to find x^* such that:

$$\lambda(x^*) = \max_{x \in L} \{\lambda(x) = f(x)/g(x)\}$$

We observe that by some transformations we could specialize the *zero-one fractional programming* problem mentioned above to the problem of finding the critical interval set.

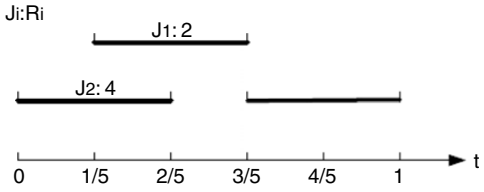

Fig. 2. A simple instance where EDF policy does not work.

Firstly, we specialize $f(x)$ and $g(x)$ a bit as follows by letting $B = \emptyset$ in $g(x)$ and incorporating the second term into the first term in $f(x)$. Therefore, we only require P to be non-empty in the following formulation.

$$f(x) = \sum_{P \in A} a_P \prod_{i \in P} x_i$$

$$g(x) = \sum_{i=1}^{n} b_i x_i$$

Given a minimum energy scheduling problem instance with jobs $J = \{J_1, J_2, \ldots, J_n\}$ and corresponding pieces $\{p_1, p_2, \ldots, p_m\}$, we do the following mapping. For each piece p_i, we map it to a variable x_i and let b_i be equal to the length of p_i. For each job J_k, we construct one element P in A which consists of all the pieces contained by the job and let $a_P = R_k$. Then $x_i = 1$ means piece p_i is chosen.

After the transformation, the problem of finding a set of intervals with maximum density becomes a special case of zero-one fractional programming in [15]. Therefore, the set of intervals with maximum density can be computed in polynomial time since zero-one fractional programming problem can be solved in polynomial time [15].

3.2. Scheduling the critical interval set

Even if the *critical interval set* of some cases is found, we may find that the commonly used scheduling policy EDF, which can guarantee feasibility in the single interval setting, no longer guarantees a feasible schedule in the multiple interval setting. A sample instance with two jobs is shown in Fig. 2. The parameters are set as $T_1 = [\frac{1}{5}, \frac{3}{5}], R_1 = 2, T_2 = [0, \frac{2}{5}] \cup [\frac{3}{5}, 1]$, $R_2 = 4$. We could easily find the *critical interval set* $T^* = [0, 1]$, the *critical group* $J_{T^*} = \{J_1, J_2\}$, and the *intensity* $g(T^*) = 6$. There are two possible interpretation of EDF in the multiple interval setting. Firstly, if we treat the latest time a job can be done as its deadline, then according to EDF, J_2 should be executed for $\frac{1}{5}$ unit of time first, then J_1 for $\frac{1}{3}$ unit. And then there is not enough time left for executing the remaining workload of J_2 using the speed 6 in $[\frac{3}{5}, 1]$. Secondly, if we treat the ending point of the current active interval as the deadline of a job, then J_2 will be executed in $[0, \frac{2}{5}]$, which makes it impossible for J_1 to be finished in its remaining time $[\frac{2}{5}, \frac{3}{5}]$.

We find a method to solve this problem based on *Linear Programming* (A similar one is first proposed by [16] for a different problem setting). Given a job set J with the *critical interval set* T^*, the *critical group* J_{T^*}, and the *intensity* $g(T^*)$, suppose that $J_{T^*} = \{J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(k)}\}$. And T^* is divided into m' pieces $p'_1, p'_2, \ldots, p'_{m'}$ using arrival times and deadlines of jobs in J_{T^*}. Suppose the length of the piece p'_j is l_j.

We can formulate the scheduling problem in the critical interval set T^* into the following linear program:

$$\begin{aligned}
&\max 1 \\
&\sum_{j=1}^{m'} x_{i,j} = \tfrac{R_i}{g(T^*)}, \quad && i=1, 2, \ldots, k \\
&\sum_{i=1}^{k} x_{i,j} = l_j, && j=1, 2, \ldots, m' \\
&x_{i,j} \geqslant 0, && \text{if } T_{\sigma(i)} \cap p'_j \neq \emptyset \\
&x_{i,j} = 0, && \text{if } T_{\sigma(i)} \cap p'_j = \emptyset
\end{aligned}$$

Here, $x_{i,j}$ represents the length of time assigned to $J_{\sigma(i)}$ in piece p'_j. Because $g(T^*)$ is the maximum speed needed in the optimal schedule, the above LP will have a feasible solution.

3.3. Optimality proof

In [7], Bansal et al. proved the optimality of YDS proposed in [1] which deals with tasks with single active interval. They gave an elegant proof by modeling the optimization problem as a convex program. Since our problem also fits into their convex program formulation, their optimality proof can also be used for our problem. We briefly state the proof as follows.

Theorem 1. *Algorithm 1 can compute the optimal schedule in polynomial time.*

Proof. We first show how to express the min-energy scheduling problem for jobs with multiple active intervals as a convex program, and then show that the schedule computed by Algorithm 1 is exactly the solution to this convex program.

To state the energy minimization problem as a convex program, we break time into intervals (pieces) $p_1, p_2, \ldots, p_m$ at release times and deadlines of the tasks $t_0, \ldots, t_m$ as shown in Section 3.1 (Any time period where no jobs are available will be removed before we break the time into pieces and this will not affect the optimality of schedules). Let $J(i)$ be the tasks that can be executed during the time interval $p_i = [t_i, t_{i+1}]$, and $J^{-1}(j)$ be the index of intervals during which task J_j can be feasibly executed. We introduce a variable $w_{i,j}$, for $J_j \in J(i)$, that represents the work done on task J_j during time $[t_i, t_{i+1}]$. The convex program is then:

$$\min E \tag{1}$$

$$R_j \leq \sum_{i \in J^{-1}(j)} w_{i,j} \quad j=1, 2, \ldots, n \tag{2}$$

$$\sum_{i=1}^{m} \left(\frac{\sum_{J_j \in J(i)} w_{i,j}}{t_{i+1} - t_i} \right)^{\alpha} (t_{i+1} - t_i) \leq E \tag{3}$$

$$w_{i,j} \geq 0 \quad i=1, 2, \ldots, m, J_j \in J(i) \tag{4}$$

It is easy to verify that program is convex. By applying KKT conditions to this program (for more details, please refer to [7]), we can conclude that a sufficient condition for a primal feasible solution to be optimal is that:

- For each task J_j, the processor runs at the same speed, say s_j, in all intervals p_i in which task J_j is run.
- The processor runs at speed no less than s_j during intervals p_i where $J_j \in J(i)$ and task J_j is not run.

Algorithm 1 obviously has these properties and hence is optimal.

4. Conclusion

This paper mainly investigates the min-energy scheduling problem of jobs with multiple active intervals. Different from the previous investigations, in our problem, the partial work done in one interval remains valid and the job only needs to complete the remaining workload in its subsequent active intervals in order to be completed. We show that this problem is polynomial time solvable using a similar algorithmic framework proposed by [1] for single interval jobs, while new techniques need to be used in the major two phases of the algorithm.

Acknowledgements

The first and the second author were supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [project No. CityU 117408]. The third author was supported by National Natural Science Foundation of China (grant no. 60775037) and the National High Technology Research and Development Program of China (863 Program) (grant no. 2009AA01Z123).

References

[1] F. Yao, A. Demers, S. Shenker, A scheduling model for reduced CPU energy, in: Proceedings of the 36th Annual Symposium on Foundations of Computer Science, FOCS, 1995, pp. 374–382.
[2] M. Li, A.C. Yao, F.F. Yao, Discrete and continuous min-energy schedules for variable voltage processors, in: Proceedings of the National Academy of Sciences of USA, vol. 103, 2006, pp. 3983–3987.
[3] B. Simons, M. Sipser, On scheduling unit-length jobs with multiple release time/deadline intervals, Operations Research 32 (1) (1984) 80–88.
[4] C.S. Shih, J.W.S. Liu, I.K. Cheong, Scheduling jobs with multiple feasible intervals, in: Proceedings of IEEE Real-Time and Embedded Computing Systems and Applications, 2003, pp. 53–71.
[5] J.J. Chen, J. Wu, C.S. Shih, T.W. Kuo, Approximation algorithms for scheduling multiple feasible interval jobs, in: Proceedings of RTCSA'05, 2005, pp. 11–16.
[6] J.J. Chen, J. Wu, C.S. Shih, Approximation algorithms for scheduling real-time jobs with multiple feasible intervals, in: Real-Time Systems, 2006, pp. 155–172.
[7] N. Bansal, T. Kimbrel, K. Pruhs, Speed scaling to manage energy and temperature, Journal of the ACM 54 (1) (2007) Article No. 3.
[8] G. Quan, X.S. Hu, Energy efficient fixed-priority scheduling for hard read-time systems, in: Proceedings of the 36th Conference on Design Automation, 1999, pp. 134–139.
[9] H.S. Yun, J. Kim, On energy-optimal voltage scheduling for fixed-priority hard real-time systems, ACM Transactions on Embedded Computing Systems 2 (3) (2003) 393–430.
[10] W. Kwon, T. Kim, Optimal voltage allocation techniques for dynamically variable voltage processors, in: Proceedings of the 40th Conference on Design Automation, 2003, pp. 125–130.
[11] M. Li, F. Yao, An efficient algorithm for computing optimal discrete voltage schedules, SIAM Journal on Computing 35 (3) (2005) 658–671.
[12] H.L. Chan, W.T. Chan, T.W. Lam, L.K. Lee, K.S. Mak, P.W.H. Wong, Energy efficient online deadline scheduling, In: Proceedings of the Eighteenth Annual ACM–SIAM Symposium on Discrete Algorithms, 2007 pp. 795–804.
[13] N. Bansal, H.L. Chan, T.W. Lam, L.K. Lee, Scheduling for speed bounded processors, in: Proceedings of the 35th International Colloquium on Automata, Languages and Programming, 2008, pp. 409-420.
[14] N. Bansal, H.L. Chan, K. Pruhs, Speed scaling with an arbitrary power function, in: Proceedings of the Twentieth Annual ACM–SIAM Symposium on Discrete Algorithms, 2009, pp. 693–701.
[15] G. Gallo, M.D. Grigoriadis, R.E. Tarjan, A fast parametric maximum flow algorithm and applications, SIAM Journal on Computing 18 (1989) 30–55.
[16] R. Sugihara, R.K. Gupta, Scheduling under location and time constraints for data collection in sensor networks, in: 28th IEEE Real-Time Systems Symposium, RTSS (work-in-progress session), Tucson, Dec. 2007.