고정 우선순위 경성 실시간 시스템에 대한 최적의 전압 스케쥴링

(On Energy-Optimal Voltage Scheduling for Fixed-Priority Hard Real-Time Systems)

윤 한 샘 † 김 지 홍 ††
(Han-Saem Yun) (Jihong Kim)

요 약 본 논문에서는 고정 우선순위 경성 실시간 시스템에 대한 에너지 측면에서의 최적의 전압 스케쥴링 문제를 고려한다. 먼저, 이 문제가 NP-hard임을 증명한다. 다음으로 이 문제에 대한 fully polynomial time approximation scheme(FPTAS)을 제시한다. 제안한 FPTAS는 주어진 임의의 $\varepsilon>0$에 대해 에너지 소모량이 최적의 전압 스케쥴에 비해 $(1+\varepsilon)$배 이내에 있는 전압 스케쥴을 문제의 입력의 크기와 $1/\varepsilon$의 다항함수 이내의 시간에 계산해준다. 실험 결과, 제안된 FPTAS는 기존의 휴리스틱에 비해 더 효율적인 전압 스케쥴을 더 빠른 시간에 찾아주었다.

키워드 : 가변 전압 프로세서, 동적 전압 조절, 실시간 시스템

Abstract We address the problem of energy-optimal voltage scheduling for fixed-priority hard real-time systems. First, we prove that the problem is NP-hard. Then, we present a fully polynomial time approximation scheme (FPTAS) for the problem. For any $\varepsilon>0$, the proposed approximation scheme computes a voltage schedule whose energy consumption is at most $(1+\varepsilon)$ times that of the optimal voltage schedule. Furthermore, the running time of the proposed approximation scheme is bounded by a polynomial function of the number of input jobs and $1/\varepsilon$. Experimental results show that the approximation scheme finds more efficient voltage schedules faster than the best existing heuristic.

Key words : variable voltage processor, dynamic voltage scaling, real-time systems

1. 서 론

정보통신 기술이 발달함에 따라 휴대전화나 개인용 휴대 단말기(PDA), 그리고 동영상 휴대전화와 같은 이동용 시스템(mobile system)의 중요성은 날로 증가하고 있다. 한 번 배터리를 충전했을 때 얼마나 오랫동안 사용할 수 있는지를 나타내는 연속 동작 가능 시간은 상업적인 이동용 시스템에 있어서 가장 중요한 성능 척도의 하나이며, 이동용 시스템의 전력 소모를 줄이는 것은 VLSI 시스템 설계에서 중요한 요소로 부각되고 있다. 특히, VLSI 시스템이 고성능, 고집적화 되어감에 따라 전력 소모는 지수적으로 증가하는 반면에 배터리의 전력 용량은 산술적인 증가에 그치고 있어서, 배터리 자체 개량보다는 VLSI 시스템의 전력 소모를 줄이는 저전력 기법이 중점적으로 연구되는 추세이다. 고성능 마이크로프로세서와 같은 비이동용 시스템에 있어서도 전력 소모는 중요한 설계 목표의 하나가 된다. 높은 전력 소모는 VLSI 내부에서 많은 열을 발생시켜 반도체의 성능 저하나 기능 이상, 심지어는 고장을 일으키기도 하기 때문이다. 이러한 문제점들 때문에 최근 들어 소프트웨어 및 하드웨어 측면에서 다양한 저전력 기법이 요구되고 있다.

일반적인 VLSI 시스템의 전력 소모는 CMOS 회로의 동적 전력 소모(dynamic power)가 대부분이며, 이때의 전력 소모 E는 $E \propto C_L \cdot N_{cycle} \cdot V_{DD}^2$ [1]로 주어진다. 여기서 C_L은 CMOS 회로의 부하 커패시턴스(load capacitance), N_{cycle}은 프로그램이 실행된 사이클 수, 그리고 V_{DD}는 공급 전압(supply voltage)을 의미한다. 전력 소모 E가 V_{DD}의 제곱에 비례하기 때문에 공급 전압 V_{DD}를 낮추는 것은 전력 소모를 줄이는 데 매우 효

· 본 연구는 대학 IT연구센터 육성지원사업의 연구결과로 수행되었음
† 비 회 원 : 서울대학교 컴퓨터공학부
hsyun@davinci.snu.ac.kr
†† 종신회원 : 서울대학교 전기컴퓨터공학부 교수
jihong@davinci.snu.ac.kr
Corresponding author임
논문접수 : 2004년 3월 9일
심사완료 : 2004년 7월 12일

과적인 방법이다. 그러나 공급 전압을 낮추면 클록 속도도 낮아지는 데, 그 이유는 CMOS의 회로 지연 시간 T_D가 $V_{DD}/(V_{DD}-V_T)^{\alpha}$에 비례하기 때문이다. 여기서 V_T는 임계 전압, α는 속도 포화 계수(velocity saturation index)[1]를 나타낸다.

실시간 시스템(real-time system)의 경우, 시스템의 최대 클럭 속도는 모든 태스크를 주어진 마감 시간(deadline)이내에 끝낼 수 있는 클럭 속도보다 크거나 같아야 실시간 동작을 보장할 수 있다. 이때, 각 태스크의 동작 상태를 살펴가면서 마감 시간 제약 조건을 만족하는 가장 낮은 클럭 속도까지 시스템의 클럭 속도를 조절해가면서 낮출 수 있으며, 클럭 속도에 따른 공급 전압도 함께 낮추어서 전력 소모를 크게 줄일 수 있다. 이것이 전압 스케쥴링(dynamic voltage scaling: DVS) 기법의 기본 개념이다. 예를 들어 그림 1(a)와 같이 어떤 태스크가 50MHz의 클럭 속도와 5.0V의 공급전압을 가진 프로세서 상에서 실행된다고 가정하자. 만약 이 태스크가 실행되는데 5×10^5사이클이 걸리고 마감 시간 조건이 25msec이라면, 프로세서는 주어진 태스크를 10msec만에 끝낼 수 있고 태스크의 마감 시간까지 15msec의 유휴 시간(idle time)을 가지게 된다. 그러나 그림 1(b)에서와 같이 클록 속도와 공급 전압이 20MHz와 2.0V로 낮아지면 유휴 시간 없이 주어진 태스크를 마감 시간에 맞추어서 끝낼 수 있고 전력 소모는 $2.0^2/5.0^2$ = 16%로 낮아지게 된다.

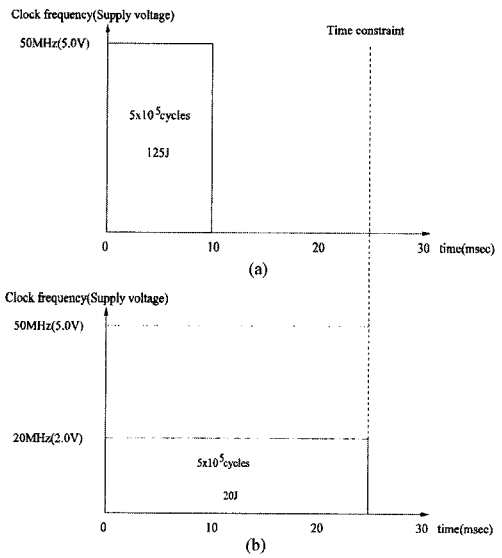


그림 1 전압 조정의 예; (a) 125J의 전력을 소비하며 15msec의 유휴시간을 가지는 경우, (b) 20J의 전력을 소비하며 유휴시간이 없는 경우

전력 소모가 특히 중요한 이동용 내장형 시스템의 시장이 급속히 커짐에 따라 여러 상업용 가변 전압 프로세서가 개발되었다(예: Intel의 Xscale [2], AMD의 K6-2+ [3], Transmeta의 Crusoe [4] 프로세서). 가변 전압 프로세서를 위해 여러 전압 스케쥴링 알고리즘들이 개발되었고 특히 경성 실시간 시스템에 대한 연구가 많이 이루어져 왔다. 경성 실시간 시스템의 경우 전압 스케쥴링 알고리즘의 목표는 모든 시간 제약 조건들을 만족시키면서도 에너지를 적게 소모하는 전압 스케쥴(voltage schedule)을 찾는 것이다. 전압 스케쥴은 전압(과 그에 대응되는 클럭 속도)의 시간에 대한 함수로 정의된다. 본 논문에서는 가변 전압 프로세서를 사용하는 고정 우선순위 실시간 시스템을 고려한다.

1.1 기존의 연구결과들

전압 스케쥴링 문제에 대한 기존의 연구는 주로 EDF 우선순위를 사용하는 실시간 시스템에 집중되어 왔다[5-8]. 예를 들어, 최적의 오프라인 EDF 전압 스케쥴을 찾는 문제에 대해서는 다항시간 알고리즘이 제시되었다[9]. EDF 스케쥴링 정책은 전압 스케쥴링 문제를 단순화 시키지만 RM(rate-monotonic)과 같은 고정 우선순위 스케쥴링 정책이 적은 오버헤드 및 예측가능성 등의 이점으로 인해 더 널리 사용되고 있다.

고정 우선순위 실시간 시스템에 대해 여러 온라인 알고리즘들[8,10,11] 및 오프라인 알고리즘들[10,12,-14]이 개발되었으나 최적의 오프라인 전압 스케쥴링 문제에 대한 연구는 거의 이루어지지 않았다. 즉, 다항시간에 수행되는 최적의 알고리즘과 문제에 대한 계산복잡도(computational complexity) 모두 알려지지 않고 있다. 최적의 고정 우선순위 전압 스케쥴링에 대해 현재까지 알려진 의미있는 연구결과는 Quan과 Hu에 의해 개발된 소모적인(exhaustive) 알고리즘[13]이 전부이다. 더욱이, Quan과 Hu는 그들의 소모적인 방법을 정당화시킬 수 있는 계산복잡도를 밝히지 못했다. Quan의 알고리즘의 복잡도는 O(N!)보다 높으므로 대부분의 실시간 시스템에 대해 사용되기 힘들다. Quan과 Hu는 다항시간에 수행되는 휴리스틱[12]도 제시하였는데, 이 문제에 대해 가장 성능이 좋은 것으로 알려져 있다. 이 휴리스틱은 효율적이기는 하나 계산된 전압 스케쥴의 에너지 측면에서의 성능을 보장할 수 없다.

1.2 연구의 의의

본 논문에서는 고정 우선순위 경성 실시간 시스템에 대한 최적의 오프라인 전압 스케쥴링 문제에 대해 이론적, 실용적으로 완전한 연구결과를 제시한다. 이 문제는 Yao에 의해 풀린 최적의 EDF 전압 스케쥴링 문제[9]에서 우선순위 할당이 EDF로부터 임의의 고정 우선순위로 일반화되었다는 점을 제외하고는 동일하다. 하지만

Quan과 Hu에 의해 예증된 것과 같이[12] 고정 우선순위 전압 스케쥴링 문제가 훨씬 더 어려운데 이는 주로 더 복잡해진 선점(preemption) 관계에 기인한다.

먼저, 최적의 고정 우선순위 전압 스케쥴링 문제가 NP-hard임을 증명하는데 이는 다항시간에 수행되는 최적의 알고리즘이 존재할 가능성이 극히 희박함을 의미한다. 다음으로, 이 문제에 대한 *fully polynomial time approximation scheme*(FPTAS)를 제시한다. FPTAS는 임의의 $\varepsilon(>0)$를 추가의 입력으로 받아 비용이 최적의 해와 비교해 $(1+\varepsilon)$배 내에 있는 근사해를 문제의 입력의 크기와 $1/\varepsilon$의 다항함수내의 시간에 계산해준다[15]. 문제가 NP-hard임을 고려할 때 제안하는 FPTAS는 이론적, 실용적 모든 측면에서 최선책으로 볼 수 있다. 제안하는 FPTAS는 근사해를 다항시간 이내에 계산 할 수 있고 ε를 조정함으로써 최적의 해와의 오차가 무한히 작아지도록 할 수 있다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 문제를 정의하고 이어 3장에서는 문제에 대한 몇가지 중요한 성질들을 분석한다. 이 결과를 바탕으로 4장에서는 문제의 계산복잡도에 대해 논하고 5장에서는 문제에 대한 FPTAS를 제시한다. 6장에서는 FPTAS를 기존의 알고리즘과 비교한 실험결과를 보여주며 7장에서는 결론과 향후 연구에 대한 방향을 제시한다.

2. 문제의 정의

문제의 입력은 임의의 고정 우선순위를 가지는 실시간 작업들의 집합 $\mathcal{J}=\{J_1, J_2, \cdots, J_{|\mathcal{J}|}\}$으로 구성되며 우선순위가 높은 순서로 정렬되어 있다. 즉, J_1이 우선순위가 가장 높고 $J_{|\mathcal{J}|}$이 우선순위가 가장 낮다. 각 작업 $J\in\mathcal{J}$는 다음과 같은 오프라인에 알려진 시간적 특성들을 가지고 있다.

- r_J : J가 배포되는 시간
- d_J : J의 데드라인
- c_J : J를 수행하는데 필요한 CPU 싸이클 수

작업 J의 우선순위는 p_J로 표기하는데 J와 J'에 대해 $p_J < p_{J'}$가 성립하면 J의 우선순위가 J'보다 높은 것으로 가정한다. 앞으로 혼란이 없는 경우에는 아래첨자로 J_i 대신에 i를 사용하여 표기를 간단하게 한다 (예: r_i , d_i , c_i는 각각 r_{J_i} , d_{J_i} , c_{J_i}를 나타낸다). 이와 같은 작업 모델은 가장 일반적인 형태인데 예를 들어 임의의 주기적(periodic) 실시간 시스템의 경우 하이퍼주기(hyperperiod) 내의 모든 태스크 객체들을 고려함으로써 이 작업 모델을 적용할 수 있다.

프로세서의 속도와 공급 전압은 일대일대응 관계에 있으므로 프로세서의 속도 $S(t)$를 사용하여 전압 스케쥴을 표기한다. 주어진 전압 스케쥴 $S(t)$로 수행시켰을 때 모든 작업들이 데드라인 내에 종료되면 $S(t)$는 유효하다고 부른다. ($S(t)$가 유효하기 위한 필요충분조건은 2.1절에서 논의된다.) 이 문제와 관련된 다른 연구[9, 12, 13]에서와 같이 프로세서의 속도는 연속적으로 변할 수 있고 시간 및 에너지 오버헤드는 없다고 가정한다. 단위시간당 에너지 소모량인 전력 소모 P는 프로세서 속도에 대한 볼록(convex)함수인데 주어진 전압 스케쥴 $S(t)$에 대해 전력 소모는 시간의 함수인 $P(S(t))$로 나타낼 수 있다.

전압 스케쥴링 문제의 목표는 에너지 소모를 최소로 하는 유효한 $S(t)$를 찾는 것인데 에너지 소모는 다음과 같이 주어진다.

$$E(S) = \int_{t_s}^{t_f} P(S(t))\,dt$$

앞으로 $\mathcal{J}$에 대한 최적의 전압 스케쥴을 $S_{opt}^{\mathcal{J}}$로 표기한다.

2.1 유효성에 대한 분석

이 절에서는 주어진 전압 스케쥴이 유효하기 위한 필요충분조건을 유도한다. 먼저 필요한 표기와 정의에 대해 설명한다.

$W(S,[t_1,t_2])$는 $S(t)$로 수행할 때 구간 $[t_1,t_2]$내에서 수행되는 CPU 싸이클 수를 나타낸다. 즉, $W(S,[t_1,t_2]) = \int_{t_1}^{t_2} S(t)\,dt$이다. $R_{\mathcal{J}}, D_{\mathcal{J}}, T_{\mathcal{J}}$는 각각 $\{r_J \mid J\in\mathcal{J}\}$, $\{d_J \mid J\in\mathcal{J}\}$, $R_{\mathcal{J}}\cap D_{\mathcal{J}}$를 나타낸다. $\mathcal{J}'\subseteq\mathcal{J}$에 대해 $C(\mathcal{J}')$는 $\sum_{J\in\mathcal{J}'} c_J$를 나타낸다. $T^{\mathcal{J}}$는 $[r_1,d_1]\times[r_2,d_2]\times\cdots\times[r_{\mathcal{J}},d_{\mathcal{J}}]$를 나타낸다. 주어진 $S_1, S_2, \cdots, S_n$이 $\forall 1\le i\le n$, $t\notin[\alpha_i,\beta_i]$ $S_i(t)=0$ $\wedge$ $\forall 1\le i\le n$, $\beta_i\le\alpha_{i+1}$ 을 만족시킬 때 $S_1, S_2, \cdots, S_n$의 연결은 다음과 같이 정의된다.

$$\oplus_{i=1}^{n} S_i = S_1\oplus S_2\oplus\cdots\oplus S_n = \sum_{i=1}^{n} S_i(t)$$

각 작업들은 수행되기 전에 배포되어야 하므로 $S(t)$는 $W(S,[0,t])\le C(\{J \mid r_J < t\})$을 만족시킨다고 가정한다.

주어진 전압 스케쥴 $S(t)$가 유효하기 위한 필요충분조건은 다음과 같다(증명은 [16]의 Theorem 1을 참조).

조건 I

$\exists(f_{J_1}, f_{J_2}, \cdots, f_{J_{|\mathcal{J}|}}) \in T^{\mathcal{J}}$

$\forall 1\le i\le|\mathcal{J}|$ $\forall r\in\{t \mid t\in R_{\mathcal{J}} \wedge t < f_{J_i}\}$

$W(S,[r,f_{J_i}]) \ge C(\{J \mid p_J \le p_{J_i} \wedge r_J\in[r,f_{J_i})\})$

주어진 $\mathcal{J}$에 대해 임의의 $J, J'\in\mathcal{J}$ $(p_J < p_{J'})$가 $d_J\le d_{J'}$ 또는 $d_{J'}\le r_J$를 만족시킬 때 $\mathcal{J}$는 EDF 우

선순위를 따르고 이를 EDF 작업집합으로 부른다. $\mathcal{J}$가 EDF 우선순위를 따를 때 조건 I은 다음과 같이 단순화된다(증명은 [16]의 Lemma 2를 참조).

조건 II

$\forall r \in R_{\mathcal{J}}, d \in D_{\mathcal{J}}\ (r < d)$

$$W(S,[r,d]) \geq C(\{J \mid [r_J, d_J] \subseteq [r,d]\})$$

조건 I, II에서 관찰할 수 있는 것처럼 고정 우선순위 전압 스케줄링 문제의 해답공간(solution space)이 EDF 전압 스케줄링 문제의 해답공간보다 훨씬 복잡하게 주어지고 이와 같이 복잡한 해답공간이 문제를 어렵게 만든다.

3. 최적해에 대한 분석

이 장에서는 유효한 전압 스케줄이 최적해가 되기 위한 성질들을 분석한다. 분석 결과는 이후에 NP-hardness의 증명과 FPTAS의 설계시 이용된다. 조건 I로 주어진 문제의 해답공간은 직접 탐색하기에는 적절하지 않을 정도로 복잡하게 주어져 있는데 이를 조건 II를 이용해 간접적으로 탐색할 수 있도록 한다.

3.1 EDF-동등성과 $|\mathcal{J}|$-튜플

조건 I과 II의 연결고리를 기술하기 위해 다음과 같이 EDF-동등성에 대해 정의한다. 주어진 $|\mathcal{J}|$-튜플 $f = (f_1, f_2, \cdots, f_{|\mathcal{J}|}) \in T^{\mathcal{J}}$에 대해 $\mathcal{J}^f$는 $\{J_1', J_2', \cdots, J_{|\mathcal{J}|}'\}$를 나타내는데 이때 J_i'의 시간적 특성은 $p_{J_i'} = p_{J_i}$, $c_{J_i'} = c_{J_i}$, $r_{J_i'} = r_{J_i}$, $d_{J_i'} = f_{J_i}$로 주어진다. 즉, 데드라인만 d_{J_i}에서 f_{J_i}로 앞당겨지고 나머지는 동일하다. 주어진 $|\mathcal{J}|$-튜플 f에 대해 $\mathcal{J}^f$가 EDF 우선순위를 따르면 f는 EDF-정렬되었다고 하고 $\mathcal{J}^f$는 $\mathcal{J}$에 대해 EDF-동등하다고 한다. 다음 정리는 유효한 전압 스케줄을 EDF-동등성을 이용해 얻을 수 있는 방법을 기술한다(증명은 [16]의 Lemma 3, 4와 Theorem 5를 참조).

정리 1. 주어진 작업집합 $\mathcal{J}$에 대해 모든 유효한 전압 스케줄의 집합을 $F_{\mathcal{J}}$로 나타내고 $\mathcal{J}$의 모든 EDF-정렬된 $|\mathcal{J}|$-튜플들의 집합을 T_{EDF}로 나타낼 때, $F_{\mathcal{J}} = \bigcup_{f \in EDF} F_{\mathcal{J}^f}$이다.

정리 1에서 알 수 있듯이 $\mathcal{J}$에 대해 유효한 전압 스케줄들과 $\mathcal{J}$에 EDF-동등한 작업집합들에 대해 유효한 전압 스케줄들은 일대일대응 관계에 있다. EDF-동등한 작업집합 $\mathcal{J}^f$에 대한 최적의 전압 스케줄 $S_{opt}^{\mathcal{J}^f}$은 Yao의 알고리즘[9]에 의해 다항시간에 쉽게 계산될 수 있으므로 $\mathcal{J}$에 대한 최적의 전압 스케줄을 찾는 문제는 $\mathcal{J}$에 EDF-동등한 작업집합들 중 $E(S_{opt}^{\mathcal{J}^f})$를 최소로 하는 $\mathcal{J}^f$를 찾는 문제로 환원시킬 수 있다.

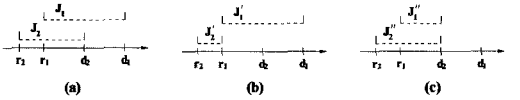


그림 2 EDF-동등한 작업집합들의 예

그림 2는 EDF-정렬된 $|\mathcal{J}|$-튜플들과 EDF-동등한 작업집합들을 예시한다. 그림 2(a)는 작업집합 $\mathcal{J} = \{J_1, J_2\}$를 보여주는데 J_2는 J_1보다 우선순위가 낮으나 데드라인이 빠르므로 $\mathcal{J}$는 EDF 우선순위를 따르지 않는다. 그림 2(b)와 (c)는 $\mathcal{J}$에 대해 EDF-동등한 두 작업집합들을 보여주는데 $\{J_1', J_2'\}$과 $\{J_1'', J_2''\}$은 각각 (d_{J_1}, r_{J_1})과 (d_{J_2}, d_{J_2})를 EDF-정렬된 $|\mathcal{J}|$-튜플로 적용시켜 얻을 수 있다. 두 작업집합 모두 EDF 우선순위를 따르므로 최적의 전압 스케줄은 Yao의 알고리즘[9]을 이용해 얻을 수 있다(뒤에서 설명되는데, $\mathcal{J}$에 대한 최적의 전압 스케줄은 c_{J_1}과 c_{J_2}에 따라 $S_{opt}^{\{J_1', J_2'\}}$ 또는 $S_{opt}^{\{J_1'', J_2''\}}$로 결정된다).

문제를 더욱 간단하게 만들기 위해 EDF-정렬된 $|\mathcal{J}|$-튜플(동등하게, EDF-동등한 작업집합)들의 탐색공간을 축소하는 것을 고려한다. 먼저, EDF-정렬된 $|\mathcal{J}|$-튜플 $f = (f_1, f_2, \cdots, f_{|\mathcal{J}|})$에 대해 다른 $|\mathcal{J}|$-튜플 $f' = (f_1', f_2', \cdots, f_{|\mathcal{J}|}')$ $(\neq f)$이 존재하여 $f_i \leq f_i'$ $(1 \leq i \leq |\mathcal{J}|)$면 f는 고려할 필요가 없는데 $\mathcal{J}^f$에 대해 유효한 모든 전압 스케줄은 $\mathcal{J}^{f'}$에 대해서도 유효하기 때문이다. $|\mathcal{J}|$-튜플 f에 대해 이와 같은 f'가 존재하지 않으면 f(또는 $\mathcal{J}^f$)는 필수적이라고 부른다. $\mathcal{J}$에 대한 최적의 전압 스케줄은 필수적인 모든 $\mathcal{J}^f$를 열거하고 Yao의 알고리즘[9]을 이용해 $S_{opt}^{\mathcal{J}^f}$를 구해서 에너지 소모를 최소로 하는 f(또는 $\mathcal{J}^f$)를 선택함으로써 찾을 수 있다. 문제가 되는 것은 필수적인 $\mathcal{J}^f$는 지수함수 이상의 수만큼 존재한다는 것인데 이들을 조심스럽게 열거함으로써 소모성을 피하면서도 근사해를 포함하는 해답공간을 충분히 탐색할 수 있도록 한다.

3.2 $|\mathcal{J}|$-순열

주어진 $|\mathcal{J}|$-튜플이 EDF-정렬되었는지를 검사하는 일은 간단하지만 반면에 어떻게 EDF-정렬된 $|\mathcal{J}|$-튜플들을 빠짐없이 헤아릴 수 있는지는 분명하지 않다. 이 절에서는 순열에 기반해 EDF-정렬된 $|\mathcal{J}|$-튜플들을 열거하는 방법을 보인다. 주어진 $|\mathcal{J}|$-튜플 $f = (f_1, f_2, \cdots, f_{|\mathcal{J}|})$에 대해 순열 $\sigma_f : \{1,2,\cdots,|\mathcal{J}|\} \Rightarrow \{1,2,\cdots,|\mathcal{J}|\}$을 f의 각 원소들을 단조증가하는 순서로 늘어놓았을 때의 순서를 나타내도록 정의한다. 즉, $f_{\sigma_f^{-1}(1)} \leq f_{\sigma_f^{-1}(2)} \leq \cdots \leq f_{\sigma_f^{-1}(|\mathcal{J}|)}$

이 성립하도록 σ_f를 정의하고 $f_i = f_j$ $(i<j)$인 경우는 $\sigma_f(i) < \sigma_f(j)$가 성립하도록 정의한다. (앞으로 이와 같이 정의된 순열을 $|\mathcal{J}|$-순열로 부른다.) 예를 들어, $f = (f_1, f_2, f_3, f_4) = (4, 10, 2, 10)$로 주어지는 $|\mathcal{J}|$-튜플에 대해 $f_3 \le f_1 \le f_2 = f_4$가 성립하므로 $(\sigma^{-1}(1), \sigma^{-1}(2), \sigma^{-1}(3), \sigma^{-1}(4)) = (3,1,2,4)$로 정의한다(동등하게, $(\sigma(1), \sigma(2), \sigma(3), \sigma(4)) = (2,3,1,4)$)

```
1:  f_{σ^{-1}(|J|)} := d_{J_{σ^{-1}(|J|)}}
2:  for (i := |J|-1 to 1)
3:      let J^H be {J_{σ^{-1}(k)} | i < k ≤ |J| ∧ σ^{-1}(k) < σ^{-1}(i)}
4:      if (r_{J_{σ^{-1}(i)}} ≥ min({r_J | J ∈ J^H} ∪ {f_{σ^{-1}(i+1)}})) return FALSE
5:      else f_{σ^{-1}(i)} := min({f_{σ^{-1}(i+1)}, d_{J_{σ^{-1}(i)}}} ∪ {r_J | J ∈ J^H})
6:      end if
7:  end for
```

그림 3 $|\mathcal{J}|$-순열 σ에 대응되는 $|\mathcal{J}|$-튜플을 구성하는 알고리즘

$|\mathcal{J}|$-순열에 대한 유용한 성질로 주어진 임의의 $|\mathcal{J}|$-순열 σ에 대해 σ에 대응되는 필수적인 $|\mathcal{J}|$-튜플이 두개 이상 존재할 수 없다는 것이다. 즉, 각 필수적인 $|\mathcal{J}|$-튜플은 대응되는 $|\mathcal{J}|$-순열로 유일하게 결정된다. (증명은 [16]의 Lemma 7을 참조.) 그림 3은 주어진 $|\mathcal{J}|$-순열 σ에 대해 σ에 대응되는 필수적인 $|\mathcal{J}|$-튜플을 구성하는 알고리즘을 보여준다. σ에 대응되는 필수적인 $|\mathcal{J}|$-튜플이 존재하지 않을 수 있는데 이 경우에는 알고리즘은 *FALSE*를 돌려준다(정확성 증명은 [16]의 Lemma 8을 참조). 대응되는 필수적인 $|\mathcal{J}|$-튜플을 가지는 $|\mathcal{J}|$-순열을 유효하다고 부르는데 필수적인 $|\mathcal{J}|$-튜플들과 유효한 $|\mathcal{J}|$-순열들은 일대일대응 관계에 있다. 그림 4(a)는 세 개의 작업으로 이루어진 작업집합을 보여주고 그림 4(b)-(d)는 EDF-동등한 작업집합들과 대응되는 $|\mathcal{J}|$-순열들을 보여준다. 가능한 $|\mathcal{J}|$-순열의 개수는 3!(=6)개지만 이들 중 세 개만이 유효하다.

3.3 시간할당 튜플에 기반한 문제 정의

2장에서 정의한 문제의 형식화는 전압 스케줄 $S(t)$에 기반하고 있다. 이 절에서는 문제를 다른 관점에서 새롭게 정의하는데 다음과 같은 직관적인 사실을 이용한다; 최적의 전압 스케줄 $S(t)$에 대해 임의의 작업 J_i는 항상 동일한 속도로 수행된다(증명은 [16]의 Lemma 9를 참조). 이 사실에 의하면 전압 스케줄링 문제는 각 작업 J_i에 수행할 수 있는 시간 a_i를 할당하는 것을 결정하는 문제로 환원될 수 있다. 주어진 전압 스케줄 $S(t)$에 대해 대응되는 시간할당 튜플 $(a_1, a_2, \cdots, a_{|\mathcal{J}|})$은 간단하게 얻을 수 있다. 역으로, 주어진 시간할당 튜플 $A = (a_1, a_2, \cdots, a_{|\mathcal{J}|})$에 대응되는 전압 스케줄 S_A는 각 작업 J_i에 일정한 속도 c_i/a_i를 할당함으로써 얻을 수 있다. 시간할당 튜플 A에 대응되는 전압 스케줄 S_A가 유효할 때 A는 유효하다고 부른다. 시간할당 튜플 A가 유효하기 위한 필요충분조건은 다음과 같다(증명은 [16]의 Lemma 10을 참조).

조건 III

$$\exists (f_{J_1}, f_{J_2}, \cdots, f_{J_{|\mathcal{J}|}}) \in T^{\mathcal{J}}$$

$$\forall 1 \le i \le |\mathcal{J}| \quad \forall r \in \{t \mid t \in R_{\mathcal{J}} \wedge t < f_{J_i}\}$$

$$\sum_{J_k/p_k \le J_i/p_i \wedge r_k \in [r, f_i)} a_k \le f_{J_i} - r$$

비슷한 식으로 EDF 작업집합에 대해서는 다음과 같이 단순화된 조건을 얻을 수 있다.

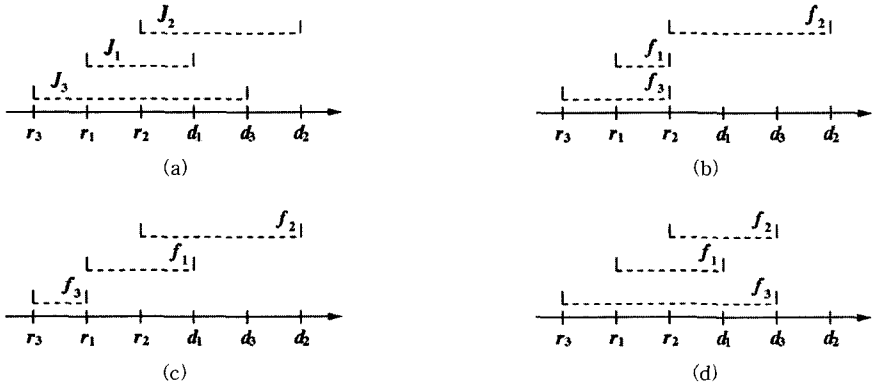


그림 4 $|\mathcal{J}|$-순열의 예;(a) 세 개의 작업으로 이루어진 작업집합과 이에 EDF-동등한 작업집합 중 $|\mathcal{J}|$-순열이 각각 (b) $(\sigma^{-1}(3), \sigma^{-1}(2), \sigma^{-1}(1)) = (2,3,1)$, (c) $(2,1,3)$, (d) $(3,2,1)$로 주어지는 것들. $(\sigma^{-1}(3), \sigma^{-1}(2), \sigma^{-1}(1)) = (1,2,3), (1,3,2), (3,1,2)$는 유효한 $|\mathcal{J}|$-순열이 아니다.)

조건 IV

$\forall r \in R_\tau, d \in D_\tau \ (r < d) \quad \sum_{J_i \in \{r_i, d_i\} \subseteq [r,d]} a_i \le d - r$

이들을 이용해 전압 스케쥴링 문제는 다음과 같이 새롭게 정의될 수 있다.

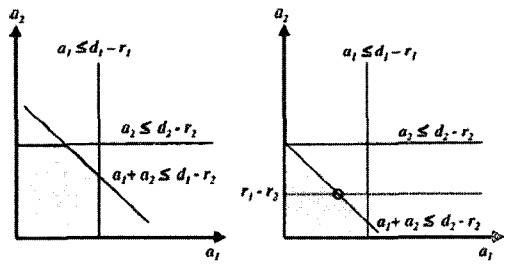
조건 III

(EDF 작업집합의 경우에는 조건 IV)를 만족시키며 에너지 소모량 $E(S_A)$를 최소로 하는 시간할당 튜플 $A = (a_1, a_2, \cdots, a_{|\tau|})$를 결정하라.

이때 에너지 소모량 $E(S_A)$는 다음과 같이 주어진다.

$$E(S_A) = \sum_{i=1}^{|\tau|} a_i \cdot P(c_i/a_i)$$

조건 III(또는 IV)로 표현되는 문제의 해답공간으로부터 문제의 복잡도에 대한 직관적인 사실을 추론할 수 있다. 조건 IV로 주어지는 EDF 작업집합에 대해서는 해답공간이 선형 부등식(linear inequality)들의 연결(conjunction)로 표현되는 반면 조건 III으로 주어지는 임의의 고정 우선순위 작업집합에 대해서는 존재적 한정사(existential quantifier)를 항상 제거할 수 없다. 결과적으로, EDF 작업집합에 대한 해답공간은 항상 볼록(convex)공간으로 주어지나 임의의 고정 우선순위 작업집합에 대해서는 항상 볼록공간으로 주어지지는 않는다.



(a) (b)

그림 5 (a) EDF 작업집합에 대한 해답공간, (b) 고정 우선순위 작업집합에 대한 해답공간

그림 5(a)와 (b)는 각각 EDF 작업집합과 고정 우선순위 작업집합의 해답공간의 예를 보여준다. 그림 5(b)는 그림 2(a)의 고정 우선순위 작업집합 $\{J_1, J_2\}$에 대한 해답공간을 나타내며 그림 5(a)는 작업집합 $\{J_1, J_2\}$과 동일하되 우선순위만 서로 바뀌어 EDF 우선순위를 따르도록 수정한 작업집합에 대한 해답공간을 나타낸다. 조건 IV로 주어지는 EDF 작업집합에 대한 해답공간은 다음과 같이 표현된다.

$a_1 \le d_{J_1} - r_{J_1} \ \wedge \ a_2 \le d_{J_2} - r_{J_2} \ \wedge \ a_1 + a_2 \le d_{J_1} - r_{J_2}$

그리고, 조건 III으로 주어지는 고정 우선순위 작업집합에 대한 해답공간은 다음과 같다.

$a_1 \le d_{J_1} - r_{J_1} \ \wedge \ a_2 \le r_{J_1} - r_{J_2}$ (그림 2(b)) $\vee$

$a_1 \le d_{J_2} - r_{J_1} \ \wedge \ a_1 + a_2 \le d_{J_2} - r_{J_2}$ (그림 2(c))

그림 5(a)와 (b)에서 EDF 작업집합과 고정 우선순위 작업집합의 해답공간은 각각 볼록(convex)공간과 오목(concave)공간으로 나타나 있다(회색으로 칠해진 지역의 각 점들은 유효한 스케쥴을 나타낸다). 일반적으로, N개의 작업들로 이루어진 EDF 작업집합의 해답공간은 $\mathbb{R}^N$에서의 볼록공간으로 주어지고 고정 우선순위 작업집합의 해답공간은 $\mathbb{R}^N$에서의 오목공간으로 주어진다. EDF 작업집합에 대해서는 볼록 최적화 기법(convex optimization technique)[17]과 같은 볼록공간에 적용할 수 있는 일반적인 최적화 기법을 이용하여 최적해를 빠른 시간에 찾을 수 있다. 하지만 오목공간에서 정의된 최적화 문제는 일반적으로 계산복잡도가 높은 것으로 알려져 있다.

4. 계산복잡도에 대한 결과

이 장에서는 최적의 고정 우선순위 전압 스케쥴링 문제의 계산복잡도에 대해 논한다. EDF 전압 스케쥴링 문제는 Yao의 알고리즘[9]으로 다항시간 이내에 풀리나 3.3절에서 암시되었듯이 고정 우선순위 전압 스케쥴링 문제는 계산복잡도가 높다. NP-complete 문제로 알려진 부분집합-합(subset-sum) 문제[18]로부터의 환원(reduction)을 통해 고정 우선순위 전압 스케쥴링 문제가 NP-hard임을 증명할 수 있다(증명은 [16]의 Theorem 11을 참조). 이에 따라 다항시간 이내에 이 문제에의 최적해를 구할 수 있는 알고리즘은 존재할 가능성이 희박하다. 하지만, NP-hardness 증명을 위해 사용된 부분집합-합 문제의 복잡성은 아주 큰 입력 숫자가 허용된다는 사실에 의존하고 있는데 이로 인해 이 문제도 강한 의미에서의 NP-hard(NP-hard in the strong sense)[18]라고 할 수 없고 보통 의미에서의 NP-hard(NP-hard in the ordinary sense)[18]일 뿐이다. 이 경우 유사다항시간(pseudopolynomial time) 알고리즘이나 FPTAS가 존재할 가능성을 배제할 수 없다. 이 문제는 실수(real number)를 다루는 최적화 문제이므로 다음 장에서는 FPTAS를 고려한다.

5. 문제에 대한 FPTAS

이 장에서는 고정 우선순위 전압 스케쥴링 문제에 대한 FPTAS를 제시한다. 먼저, 항상 최적해를 찾는 동적 프로그래밍(dynamic programming) 기법을 고려하는데 지수시간에 동작할 수 있다. 다음으로 이 동적 프로그래밍을 rounding-the-input-data 기법[15]을 이용해 FPTAS로 변환시킨다. 이 기법은 충분히 가까운 입력

자료들을 대표값으로 통일하여 사용함으로써 동적 프로그램의 수행시간을 다항시간으로 축소시킨다. FPTAS의 오차는 가까움의 정도를 어느 정도의 크기로 결정하는지에 따라 결정된다. 즉, 가까움의 정도에 대한 한계치를 작게 정함에 따라 FPTAS의 오차도 작아진다. 반면에 오차한계가 작아질수록 계산 시간은 길어지게 된다.

5.1 최적해를 계산하는 알고리즘

이 절에서는 3장에서 설명된 최적해에 대한 성질들을 이용하여 지수시간에 최적해를 구하는 알고리즘을 제시한다. 제시하는 최적의 알고리즘은 결과적으로는 모든 필수적인 EDF-동등 작업들을 열거하나 Quan과 Hu의 소모적인 알고리즘[13]과 달리 실제로 모두 열거하지 않고 모두 열거하는 효과만 내도록 한다. 더욱이, 동적 프로그래밍 형태로 설계되므로 FPTAS로 쉽게 변환될 수 있도록 한다.

문제를 동적 프로그래밍 형태로 형식화할 때 먼저 고려해야할 것은 적당한 중복되는 하위의 문제구조를 찾는 것인데 여기에 동적 프로그래밍을 반복적으로 적용시킬 수 있어야 한다. 이 문제에서는 최적의 하위의 구조(optimal substructure)가 자연스럽게 구분(blocking) 튜플에 의해 반영될 수 있는데 구분튜플은 단순히 $T^{\mathcal{J}}$의 시간들을 커지는 순서로 늘어놓은 것이다(이후에 구분튜플을 명확히 정의한다). 즉, 본래의 문제에 대한 최적해는 구분튜플에 의해 정의된 부분구간들에 대한 최적해들을 단지 합침으로써 얻을 수 있게 된다. 그림 6은 한 작업집합과 이에 EDF-동등한 작업집합을 보여주는데 그림 6(b)에서 굵은 점선들로 표시된 구분튜플 $(r_N, r_{N-3}, d_{N-1}, \cdots, r_2, d_2)$에 의해 전체 시간구간이 분할되었다. 주목할 점은 각 부분구간내의 작업집합들 역시 EDF 우선순위를 따른다는 것이다.

본래의 문제는 전체 시간구간을 부분구간들로 분할함으로써 하위의 문제들로 분할되는데 각 부분구간에 속한 작업들이 EDF 작업집합이 되도록 한다. 어떤 작업의 데드라인이 배포된 부분구간의 바깥쪽에 존재할 때 그 데드라인은 부분구간의 마지막 시점으로 변경될 수 있다. 분할된 각 부분구간은 Yao의 알고리즘[9]을 이용해서 다항시간에 최적으로 스케줄될 수 있다. 문제가 되는 것은 전체 시간구간에 대한 최적의 전압 스케줄을 만들어주는 부분구간들로의 분할(즉, 구분튜플)을 어떻게 찾는 가이다.

이 절에서는 두개의 작업집합에 대한 예를 통해서 최적의 알고리즘에 대한 직관을 먼저 설명한 후 이들을 통합하여 임의의 작업집합에 적용할 수 있는 최적의 알고리즘을 제시한다. 첫 번째 예로 간단하지만 예시에 적합한 특성을 가진 그림 6(a)의 작업집합 $\mathcal{J}=\{J_1, J_2, \cdots, J_N\}$을 이용한다. $\mathcal{J}$의 작업들은 $r_{i+1}<r_i<d_{i+1}<d_i$를 만족시키며 역-EDF 우선순위를 따르고 있음에 주목한다. $\mathcal{J}$에 대한 임의의 필수적인 EDF-동등한 작업집합 $\mathcal{J}^e$ (예를 들어 그림 6(b)의 작업집합)은 $\mathcal{J}_1^e, \mathcal{J}_2^e, \cdots, \mathcal{J}_k^e$로 분할될 수 있는데 이때 각 $\mathcal{J}_i^e$는 EDF 우선순위를 따르며 $\mathcal{J}_i^e$에 속한 작업들의 수행구간들의 합집합 I_i는 다른 I_j와 겹치지 않는다. 보다 구체적으로, 다음이 성립한다.

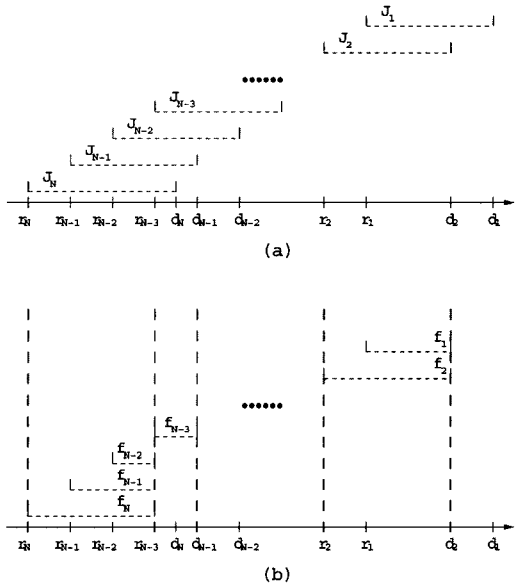


그림 6 구분튜플로 전체구간이 분할되는 예; (a) 본래의 작업집합, (b) $|\mathcal{J}|$-튜플 $f=(f_1, f_2, \cdots, f_{N-3}, f_{N-2}, f_{N-1}, f_N)=(d_2, d_2, \cdots, d_{N-1}, r_{N-3}, r_{N-3}, r_{N-3})$로 정의되는 EDF-동등한 작업집합

$$\forall 1 \le i < j < k,\ \forall J \in \mathcal{J}_i^e, J' \in \mathcal{J}_j^e,\ d_J \le r_{J'}$$

따라서, $\mathcal{J}^e$에 대한 최적의 전압 스케줄 $S_{opt}^{\mathcal{J}^e}$는 $\mathcal{J}_i^e$들에 대한 최적의 전압 스케줄들을 연결시켜 놓은 것과 같다.

$$S_{opt}^{\mathcal{J}^e}(t) = \oplus_{i=1}^{k} S_{opt}^{\mathcal{J}_i^e}(t)$$

각 $\mathcal{J}_i^e$는 EDF 우선순위를 따르므로 Yao의 알고리즘[9]에 의해 최적으로 스케줄될 수 있다.

결과적으로, 최적의 고정 우선순위 전압 스케줄링 문제는 에너지 소모를 최소로 하는 분할을 찾는 것으로 환원할 수 있는데 여기서는 구분튜플을 이용하여 분할을 정의하도록 한다. 예를 들어, 그림 6(b)와 같이 f_N

으로서 r_{N-3}을 선택한다고 할 때 $\mathcal{J}^e$가 EDF 우선순위를 따르고 필수적이 되기 위해서는 f_{N-1}과 f_{N-2}는 모두 r_{N-3}이 되어야 한다. 그림 6(b)와 같이 이 세 작업들이 r_{N-3}의 굵은 수직선에 의해 다른 작업들과 완전히 분리되어 있음을 알 수 있다. 이 세 작업들은 분할된 작업집합 $\mathcal{J}_1^e$을 이루고 나머지 작업집합들 $\mathcal{J}_2^e, \cdots, \mathcal{J}_k^e$도 같은 식으로 얻을 수 있다. 이와 같이 임의의 필수적인 EDF-동등한 작업집합은 구분튜플을 이용해 분할되고 표현될 수 있다.

구분튜플 $b=(b_1, b_2, \cdots, b_l)$ $(b_1<b_2<\cdots<b_l,\ b_j\in T_{\mathcal{J}})$이 다음을 만족하면

$$\forall 1\le j< l,\ \exists J_i,\ b_j=r_i \wedge b_{j+1}\le d_i$$

대응되는 EDF-정렬된 $|\mathcal{J}|$-튜플 $f=(f_1, f_2, \cdots, f_N)$은 다음과 같이 주어진다.

$$f_k=b_j \text{ s.t. } r_k\in[b_{j-1}, b_j) \text{ for all } 1\le k\le N$$

이와 같은 $[b_{j-1}, b_j]$을 단위구간이라 부른다. 예를 들어, 그림 6(a)에서 $[r_N, r_{N-3}]$과 $[r_N, d_N]$은 단위구간이지만 $[r_N, d_{N-1}]$은 단위구간이 아니다. (이후에 단위구간에 대해 명확히 정의한다.) t_h를 $T_{\mathcal{J}}$에서 h번째로 빠른 시점으로 두고 $S_{h,g}$를 $\mathcal{J}_{h,g}=\{J_i' \mid r_{J_i}\in[t_h, t_g)\}$에 대해 구간 $[t_h, t_g]$내에서 정의된 최적의 전압 스케쥴로 나타내자. 이때 $r_{J_i'}=r_{J_i}$, $c_{J_i'}=c_{J_i}$, $p_{J_i'}=p_{J_i}$, $d_{J_i'}=\min\{d_{J_i}, t_g\}$와 같이 J_i'을 정의한다. 그러면 다음을 얻을 수 있다.

$$E(S_{opt}^{\mathcal{J}}) = E(S_{1,|\mathcal{J}|})$$
$$= \min\{\sum_{j=1}^{h-1} E(S_{h_j,h_{j+1}}) \mid 1=h_1<h_2<\cdots<h_k=|\mathcal{J}|,$$
$$[t_{h_j}, t_{h_{j+1}}] \text{은 단위구간}\}$$

주어진 단위구간 $[t_{h_j}, t_{h_{j+1}}]$에 대해 $S_{h_j,h_{j+1}}$는 Yao의 알고리즘[9]을 이용해 얻을 수 있다. 즉, 그림 6에 주어진 작업집합에 대해서는 전압 스케쥴링 문제는 다음과 같이 정의될 수 있다.

구분튜플 $(h_1, h_2, \cdots, h_k)$ $(1=h_1<\cdots<h_k=|\mathcal{J}|)$중 $q_{h_1,h_2}+q_{h_2,h_3}+\cdots+q_{h_{k-1},h_k}$를 최소로 하는 것을 선택하라. 이때, $[t_{h_j}, t_{h_{j+1}}]$는 구분튜플이고 $q_{h_j,h_{j+1}}$은 $E(S_{h_j,h_{j+1}})$를 나타낸다(이는 Yao의 알고리즘[22]으로 구할 수 있다).

그림 6의 작업집합은 동적 프로그래밍 형태로 문제를 정형화하는 과정을 보이는데 적절하나 쉽게 분할할 수 있는 구조를 이루고 있는 것은 작업집합이 EDF 우선순위의 역순위를 따르기 때문이다. 예를 들어 그림 6(b)에서 J_{N-1}과 J_{N-2}의 수행구간 내에 r_{N-3}이 존재하고 f_N으로서 r_{N-3}이 선택되는데, EDF 우선순위를 따르기 위해서 f_{N-1}과 f_{N-2} 모두 r_{N-3}보다 클 수 없다. 우선순위가 그림 6(a)의 작업집합과 동일한 형태로 주어지지 않는다면 분할하는 문제는 복잡해진다. 예를 들어 그림 3(c)과 (d)와 같이 주어진 필수적인 EDF-동등한 작업집합들은 이와 같은 과정을 통해서 얻을 수 없다.

그림 7(a)의 작업집합에서 J_4는 우선순위가 가장 낮고 데드라인이 가장 늦은데 이로 인해 그림 7(b)-(d)의 필수적인 EDF-동등한 작업집합에서와 같이 f_4는 항상 d_4가 되어야 한다. 따라서 모든 단위구간은 J_4의 작업량 중 일부를 포함할 수밖에 없고 이것을 배경작업량(background workload)이라 부른다. 이제 위의 동적

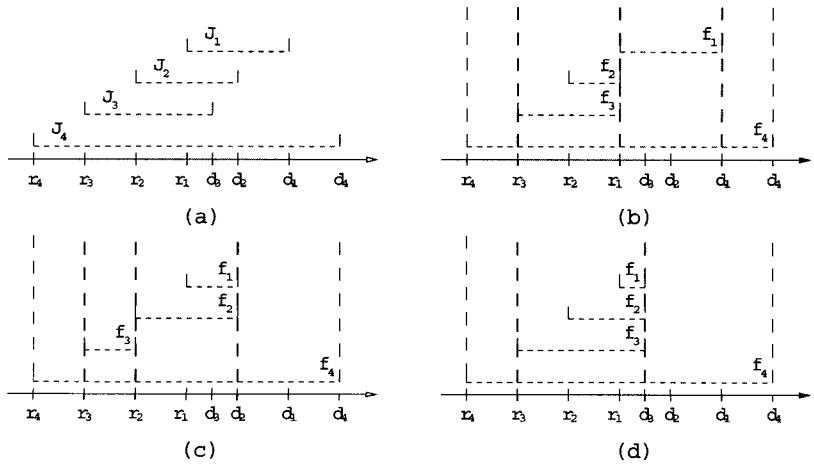


그림 7 배경작업량의 예

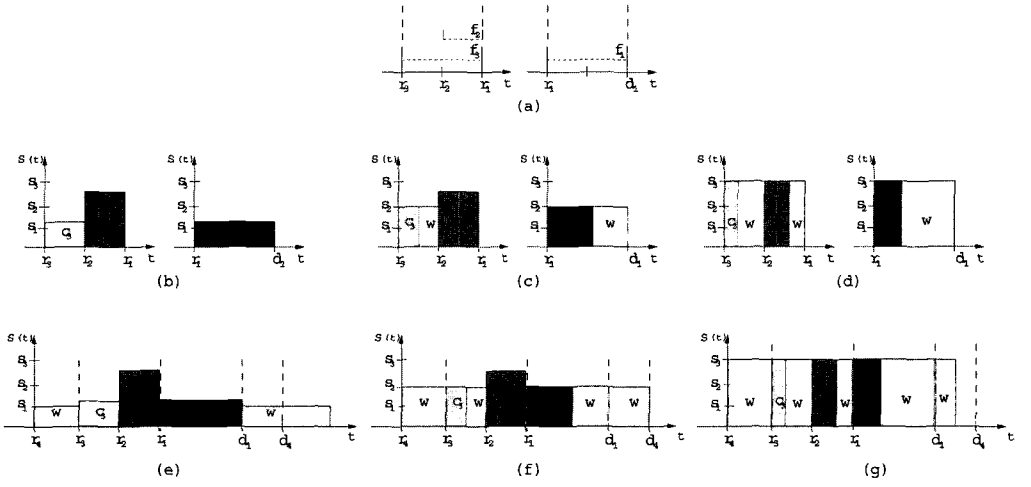


그림 8 배경작업량을 고려한 알고리즘을 보여주는 예

프로그래밍을 배경작업량을 다룰 수 있도록 확장시킨다. 앞서 설명한 바와 같이 최적의 전압 스케줄 하에서 그림 7(a)의 J_4는 항상 같은 속도로 수행된다. 당분간 그 속도가 $S_C = \{s_1, s_2, s_3\}$중 하나라고 가정하자. 먼저, 각 s_i에 대해 각 단위구간에서 수행되어야 할 J_4의 배경작업량을 구하고, 그림 6의 경우와 비슷한 방법을 통해 에너지를 최소로 하는 필수적인 EDF-동등한 작업집합을 구한다. 이때, 앞의 경우와 달리 각 구간에서의 배경작업량의 합이 J_4의 전체 작업량보다 적으면 유효한 스케줄을 얻을 수 없으므로 고려하지 않는다.

그림 8(a)는 단위구간 $[r_3, r_1]$과 $[r_1, d_1]$을 보여주는데 이들은 그림 7(b)의 작업집합으로부터 얻은 것이다. 그림 8(b)-(d)는 J_4를 각각 s_1, s_2, s_3으로 수행시켰을 때의 각 단위구간에서의 최적의 전압 스케줄을 나타낸다. 그림에서 J_1, J_2, J_3의 작업량은 각각 c_1, c_2, c_3으로 J_4의 배경작업량은 w로 표시된다. 각 단위구간에서의 각 속도에 대한 배경작업량의 크기와 이에 맞춰 결과적으로 정해지는 최적의 전압 스케줄은 Yao의 알고리즘[9]을 다음과 같이 변경시켜서 구할 수 있다. 결정적인(critical) 구간이 결정되었을 때 할당될 속도(그 구간의 세기(intensity))가 배경작업량이 수행될 속도보다 적거나 같으면 그 결정적인 구간을 포함함 모든 스케줄되지 않은 구간에 배경작업량이 수행될 속도를 할당한다. 이와 같은 방법으로 그림 8(b)-(d)에 나타난 것과 같이 각 구간에서 수행할 배경작업량의 크기를 결정할 수 있다.

각 단위 구간에 대해 배경작업량의 크기와 최적의 전압 스케줄을 구한 후에 그림 6에서와 같은 과정을 통해 전체 구간에 대한 최적의 전압 스케줄을 구한다. 배경작업량의 속도에 대해 탐색을 할 때 유효하지 않은 스케줄은 고려하지 않는다. 그림 8(e)의 전압 스케줄에서 J_4는 s_1로 수행되고 데드라인까지 수행을 완료하지 못하므로 유효하지 않고 따라서 고려하지 않는다. 그림 8(g)의 전압 스케줄은 유효하기는 하나 최적해가 아니고 그림 8(f)의 전압 스케줄은 유효한 최적의 전압 스케줄이다.

```
1:   f_{σ^-1(|J|)} := d_{J_{σ^-1(|J|)}}
2:   b_σ^w := (d_{σ^-1(|J|)})
3:   for (i := |J|-1 to 1)
4:       let J^H be {J_{σ^-1(k)} | i < k ≤ |J| ∧ σ^-1(k) < σ^-1(i)}
5:       if (r_{J_{σ^-1(i)}} ≥ min({r_J | J ∈ J^H} ∪ {f_{σ^-1(i+1)}})) return FALSE
6:       else f_{σ^-1(i)} := min({f_{σ^-1(i+1)}, d_{J_{σ^-1(i)}}} ∪ {r_J | J ∈ J^H})
7:       end if
8:       if (f_{σ^-1(i)} ≤ min{r_J | r ∈ J^H}) append f_{σ^-1(i)} onto the head of b_σ^w
9:       end if
10:  end for
11:  append min R_J onto the head of b_σ^w
```

그림 9 $|\mathcal{J}|$-순열 σ에 대응되는 구분튜플을 구성하는 알고리즘

지금까지 설명한 두 예에 대한 고찰을 바탕으로 임의의 고정 우선순위 작업집합에 적용시킬 수 있는 최적의 알고리즘을 구성한다. 먼저, 이전에 명확히 정의되지 않았던 구분튜플, 단위구간, 배경작업량에 대한 정의를 내린다. 주어진 $|\mathcal{J}|$-순열 σ에 대해 그림 9의 알고리즘은 대응되는 구분튜플 $b_\sigma = (b_1, b_2, \cdots, b_l)$ $(b_1 < b_2 < \cdots < b_l,\ b_j \in T_{\mathcal{J}})$을 구성한다. 이 알고리즘은 2, 8, 9, 11번째 줄들을 제외하고 그림 3의 알고리즘과 동일하다.

정의 1. 주어진 $|\mathcal{J}|$-순열 σ에 대해 그림 9의 알고리즘에 의해 구성된 튜플 b_σ^w를 구분튜플이라 부른다. 구간 $[t, t']$에 대해 구분튜플 $b_\sigma^w = (b_1, b_2, \cdots, b_k)$이

존재하여 $[t,t'] \equiv [b_i, b_{i+1}']$ $(\exists 1 \le i < k)$이 성립하면 $[t,t']$를 단위구간이라 부른다. 그리고, 작업집합 $\Im_{[t,t']^w} = \{J' | J \in \Im, r_J \in [t,t') \wedge (\exists J_h \in \Im, p_{J_h} < p_J \wedge r_{J_h} = t' \wedge d_J \in [r_{J_h}, d_{J_h}))\}$ ($r_{J'} = r_J$, $c_{J'} = c_J$, $p_{J'} = p_J$, $d_{J'} = \min\{d_J, t'\}$)은 구간 $[t,t']$에 의해 유도되었다고 부른다. 마지막으로, 주어진 구분튜플 $b_\sigma^w = (b_1, b_2, \cdots, b_k)$과 이에 대응되는 필수적인 EDF-동등한 작업집합 $\Im$에 대해 $\Im' - \cup_{j=1}^{k-1} \Im_{[b_j, b_{j+1}]^w}$에 속한 모든 작업들을 구분튜플 b_σ^w에 대한 배경작업(background job)이라 부르고 배경작업의 작업량을 배경작업량이라 부른다.

정의 1을 이용하여 동적 프로그래밍에 기반해서 구성한 최적의 알고리즘을 그림 10에서 보여준다. 이 알고리즘은 먼저 각 단위구간에 대해 최적의 전압 스케줄을 구한다. 이때 알고리즘은 배경작업량을 수행할 속도로 S_C에 속한 모든 값들을 고려하는데 S_C는 다음과 같이 정해진다.

$$S_C = \{ \frac{C(\Im')}{\sum_{i=0}^{k-1} (t_{p_{2i+2}} - t_{p_{2i+1}})} \mid \Im' \subset \Im,\ t_1 < t_2 < \cdots < t_{p_{2k}},\ t_j \in T_\Im \}$$

최적의 전압 스케줄 하에서 배경작업량의 속도가 반드시 S_C에 포함되는 것은 명확하다. (S_C의 크기는 다항함수로 제한되지 않아 그림 10의 알고리즘이 다항시간에 수행되지 않지만 FPTAS를 제시할 5.2절에서 S_C의 크기를 다항함수 이내로 줄이는 것에 대해 설명한다.) 각 단위구간에 대해 계산된 최적의 전압 스케줄들을 이용하여 알고리즘은 동적 프로그래밍을 이용하여 전체 구간에 대한 최적의 전압 스케줄을 탐색한다. (알고리즘의 정확성에 대한 증명은 [16]의 Lemma 17-23과 Corollary 24를 참조.) 최적의 알고리즘은 S_C의 크기로 인해 다항시간 내에 수행되지 않지만 FPTAS로 변환되기 쉬운 형태로 되어 있다.

5.2 FPTAS

이 절에서는 5.1절에서 제시한 최적의 전압 스케줄링

---

procedure OPTIMAL_VOLTAGE_SCHEDULE ($\mathcal{J}$)
/* $T_\mathcal{J} = \{t_1, t_2, \cdots, t_N\}$, $S_C := \{s_1, s_2, \cdots, s_n\}$ */
1: **foreach** ($s \in S_C$)
2: $\mathbf{V} := \{v_1, v_2, \cdots, v_N\}$
3: $\mathbf{E} := \{(v_i, v_j) \mid [t_i, t_j]$ is weakly-atomic$\}$
4: **foreach** $((v_i, v_j) \in \mathbf{E})$
5: $w((v_i, v_j)) := W(\max\{S_{\text{opt}}^{\mathcal{J}_{[t_i,t_j]}}(t), s\}, [t_i, t_j]) - W(S_{\text{opt}}^{\mathcal{J}_{[t_i,t_j]}}(t), [t_i, t_j])$ /* weight of edges */
6: **end foreach**
7: Find longest paths between all pairs of vertices in $\mathbf{V}$. /* Note that G is acyclic. */
8: **foreach** ($1 \le i < j \le N$ s.t. $[t_i, t_j]$ is a concatenation of weakly-atomic intervals)
/* The longest path from v_i to $v_j = \langle v_{q_1}, v_{q_2}, \cdots v_{q_l} \rangle$
9: c : the weight of the longest path from v_i to v_j.
10: $E_{i,j}[c] : E(\cdots, \sum_{h=1}^{l-1} \max\{ S_{\text{opt}}^{\mathcal{J}_{[q_j, q_{j+1}]}}(t), s\}, [t_i, t_j])$
11: **end foreach**
12: **end foreach**
13: **for** ($i :=$ 1 to $N-1$)
14: **for** ($j :=$ 1 to $N-i$)
15: $E_{j,j+i} : \infty^+$
16: **for** ($k :=$ $j+1$ to $j+i$)
17: $c_{j,j+i,k} : C(\{J \in \mathcal{J}^B \mid r_J \in [t_j, t_k) \wedge d_J \in [t_k, t_{j+i}]\})$
18: $E_{j,j+i,k} : E_{j,k}[c_{j,j+i,k}] + E_{k,j+i}$
19: **if** ($E_{j,j+i} > E_{j,j+i,k}$ **and** $S_{\text{opt}}^{\mathcal{J}_{[t_j,t_k]^w \cup \{J \in \mathcal{J}^B \mid [r_J, d_J] \subseteq [t_j, t_k]\}}}[c_{j,j+i,k}]$ is feasible for $\mathcal{J}_{[t_i,t_j]^w} \cup \{J \in \mathcal{J}^B \mid [r_J, d_J] \subseteq [t_i, t_j]\}$)
20: $E_{j,j+i} : E_{j,j+i,k}$, $h := k$
21: **end if**
22: **end for**
23: $\mathbf{b}_{j,j+i} : \{t_h\} \cup \mathbf{b}_{j,h} \cup \mathbf{b}_{h,j+i}$
24: **end for**
25: **end for**
/* $E_{1,N} = E(S_{\text{opt}}^{\mathcal{J}})$ and $S_{\text{opt}}^{\mathcal{J}} \equiv S_{\text{opt}}^{\cup_{h=1}^{l-1} \mathcal{J}_{[b_h, b_{h+1}]^w} \cup \mathcal{J}^B}$ where $\mathbf{b}_{1,N} = (b_1, b_2, \cdots, b_l)$ */
26: $\mathcal{J}_{\text{opt}} : \cup_{h=1}^{l-1} \mathcal{J}_{[b_h, b_{h+1}]^w} \cup \mathcal{J}^B$ where $\mathbf{b}_{1,N} = (b_1, b_2, \cdots, b_l)$
27: return $S_{\text{opt}}^{\mathcal{J}_{\text{opt}}}$ /* $\mathcal{J}_{\text{opt}}$ is an EDF job set. So, $S_{\text{opt}}^{\mathcal{J}_{\text{opt}}}$ can be directly computed by Yao's algorithm */
end procedure

그림 10 지수시간에 동작하는 최적의 전압 스케줄링 알고리즘

알고리즘을 약간 변경하여 FPTAS를 유도한다. 최적의 알고리즘의 수행시간은 지수함수로 표현되는데 이는 S_C의 크기가 지수함수로 결정되기 때문이다. 따라서 S_C의 크기를 다항함수로 줄이면서도 계산되는 전압 스케쥴이 최적해에 충분히 가깝도록 해야 한다. FPTAS에서는 다음과 같이 정의된 S_C'를 S_C 대신 사용하여 수행시간을 다항시간으로 떨어뜨린다.

$$S_C' = \{\min\{S_C\} \cdot (1+\varepsilon \cdot \rho_P)^k \mid k = 0,1,\cdots, \lceil \log_{1+\delta}(\max\{S_C\}/\min\{S_C\}) \rceil\}$$

이때, ρ_P는 $\dfrac{\log 2}{\max\{\dfrac{P(x)}{P(x)} \cdot x \mid x > 0\}}$ 를 나타낸다.

그림 11은 전압 스케쥴링 문제에 대한 FPTAS를 보여주는데 S_C 대신 S_C'를 사용하고 1, 12, 20, 21번째 라인을 제외하고 그림 10의 최적의 알고리즘과 동일하다. 이 알고리즘은 $\mathcal{J}$외에 임의의 오차한계 ε를 추가의 입력으로 받아서 최적해와 비교해 $(1+\varepsilon)$배 이내의 에너지를 소모하는 근사해를 계산해주는데 수행시간은 $|\mathcal{J}|$와 $1/\varepsilon$의 다항함수내로 제한된다(증명은 [16]의 Lemma 25와 Theorem 26을 참조).

6. 실험 결과

제안한 FPTAS의 성능을 측정하기 위해 고정 우선순위 전압 스케쥴링 문제에 대해 현재 가장 성능이 좋은 것으로 알려진 Quan과 Hu의 휴리스틱[12]을 비교대상으로 사용해 에너지 측면에서의 성능과 수행시간을 비교하였다. 실험에서 에너지 소모량은 공급 전압의 제곱에 비례한다고 가정했으며 주어진 공급 전압 V에 대해 대응되는 속도(CPU 클럭)는 $(V_{DD}-V_{TH})^{\alpha}/V_{DD}$에 비례한다고 가정하였는데, 이때 V_{TH}와 α는 각각 0.5V와

```
procedure APPROX_VOLTAGE_SCHEDULE (J, ε)
      /* T_J = {t_1, t_2, ..., t_N} */
      /* S'_C = {min{S_C}·(1+δ)^k | k = 0, 1, ..., ⌈log_{1+δ}(max{S_C}/min{S_C})⌉ where δ = ε·ρ_P */
 1:   Initialize C_{i,j} := {} for 1 ≤ i < j ≤ N.
 2:   foreach (s ∈ S'_C)
 3:       V := {v_1, v_2, ..., v_N}
 4:       E := {(v_i, v_j) | [t_i, t_j] is weakly-atomic}
 5:       foreach ((v_i, v_j) ∈ E)
 6:           w((v_i, v_j)) := W(max{S_opt^{J_{[t_i,t_j]}}(t), s}, |t_i, t_j|) − W(S_opt^{J_{[t_i,t_j]}}(t), |t_i, t_j|)   /* weight of edges */
 7:       end foreach
 8:       Find longest paths between all pairs of vertices in V.  /* Note that G is acyclic. */
 9:       foreach (1 ≤ i < j ≤ N s.t. [t_i, t_j] is a concatenation of weakly-atomic intervals)
            /* The longest path from v_i to v_j = ⟨v_{q_1}, v_{q_2}, ..., v_{q_l}⟩
10:           c := the weight of the longest path from v_i to v_j.
11:           E_{i,j}|c| := E(÷_{h=1}^{l-1} max{ S_opt^{J_{[t_{q_h}, t_{q_{h+1}}]}}(t), s}, |t_i, t_j|)
12:           C_{i,j} := C_{i,j} ∪ {c}
13:       end foreach
14:   end foreach
15:   for (i := 1 to N − 1)
16:       for (j := 1 to N − i)
17:           E_{j,j+i} := ∞^+
18:           for (k := j + 1 to j + i)
19:               c_{j,j+i,k} := C({J ∈ J^B | r_J ∈ |t_j, t_k) ∧ d_J ∈ |t_k, t_{j+i}|})
20:               c' := min{c ∈ C_{j,k} | c ≥ c_{j,j+i,k}}
21:               E_{j,j+i,k} := E_{j,k}|c'| + E_{k,j+i}
22:               if (E_{j,j+i} > E_{j,j+i,k} and
                        S_opt^{J_{[t_j,t_k]^w} ∪ [c_{j,j+i,k}]} is feasible for J_{[t_i,t_j]^w} ∪ {J ∈ J^B || r_J, d_J| ⊆ |t_i, t_j|})
23:                   E_{j,j+i} := E_{j,j+i,k} ,  h := k
24:               end if
25:           end for
26:           b_{j,j+i} := {t_h} ∪ b_{j,h} ∪ b_{h,j+i}
27:       end for
28:   end for
      /* E_{1,N} < (1 + ε)·E(S_opt^J) */
29:   J_ε := ∪_{h=1}^{l-1} J_{[b_h, b_{h+1}]^w} ∪ J^B where b_{1,N}  (b_1, b_2, ..., b_l)
30:   return S_opt^{J_ε}  /* J_ε is an EDF job set. So, S_opt^{J_ε} can be directly computed by Yao's algorithm */
end procedure
```

그림 11 최적의 전압 스케쥴링 문제에 대한 FPTAS

1.3으로 가정하였다 [15].

표 1 실시간 어플리케이션들에 대한 실험 결과

		Normalized Energy			CPU Time(s)	
Applications		MPEG4	CNC	Avionics	CNC	Avionics
#jobs		22	289	1372	289	1372
FPTAS	ε =0.1%	1	1	1	44.71	4506.63
	ε =0.5%	1.003	1.004	1.003	11.67	1021.48
	ε =1.0%	1.006	1.008	1.007	6.12	631.15
	ε =1.5%	1.012	1.013	1.011	5.16	512.32
	ε =2.0%	1.017	1.018	1.018	3.81	313.15
Quan [12]		1.041	1.062	1.059	4.76	580.32

실험을 위해 널리 사용되고 있는 실시간 어플리케이션인 MPEG4 영상전화, CNC[19], Avionics[20]의 주기적 태스크들로부터 작업집합들을 얻어내어 사용하였다. 표 1은 이에 대한 결과를 보여준다. 각 작업(즉, 태스크 객체)의 작업량은 [WCET/10,WCET] 구간에서의 Gaussian 분포를 따라 임의로 결정하였다. 에너지 값들은 ε=0.1%로 했을 때의 FPTAS가 구한 전압 스케쥴의 에너지 값을 기준으로 정규화 하였다. 표 I에서 볼 수 있듯이 FPTAS는 Quan의 휴리스틱[12]보다 빠른 시간에 에너지 소모가 더 적은 전압 스케쥴들을 계산해 주었다. 실험에서 FPTAS의 실제 오차는 오차 한계치인 ε보다 항상 작게 나옴을 확인할 수 있다(MPEG4 영상전화에 대한 CPU 시간은 생략하였는데 계산이 0.1초 내에 이루어져 정확한 측정이 힘들었기 때문이다).

실시간 어플리케이션외에도 50~1600개의 작업들을 임의로 생성하여 이들을 대상으로 실험하였다. Quan의 휴리스틱[12]의 성능에 영향을 미치는 가장 중요한 요소는 작업들 간의 간섭하는 정도라고 추정하고 이를 변화시키며 작업들을 생성하였다. 첫 번째 부류의 작업집합들(Class 1)로 각 작업의 배출시간, 수행구간의 길이, 작업량을 각각 [0,1000], [50,100], [0.2, 1.0]에서의 균일분포를 따라 임의로 결정하였다. (프로세서의 속도의 최대값은 항상 적당히 조절할 수 있으므로 작업량의 상대치만 고려해도 충분하다.) 두 번째 부류의 작업집합들(Class 2)과 세 번째 부류의 작업집합들(Class 3)은 수행구간의 길이를 [50,100]대신 각각 [100,300]과 [300, 500]에서의 균일분포를 따라 임의로 결정하였다. Class 1, Class 2, Class3은 각각 적은, 중간의, 큰 정도의 간섭을 가진 작업집합들에 대응된다. 표 2, 3, 4는 각각 Class 1, Class 2, Class 3에 대한 실험결과를 나타낸다. 표에서 볼 수 있듯이 작업들 간의 간섭 정도가 커짐에 따라 Quan의 휴리스틱[12]에 대한 FPTAS의 향상폭이 크다.

표 2 합성한 작업들(Class 1)에 대한 실험 결과

		Normalized Energy					
Jobs sets		$\mathcal{J}_1$	$\mathcal{J}_2$	$\mathcal{J}_3$	$\mathcal{J}_4$	$\mathcal{J}_5$	$\mathcal{J}_6$
#jobs		50	100	200	400	800	1600
FPTAS	ε =0.1%	1	1	1	1	1	1
	ε =0.5%	1.003	1.003	1.004	1.004	1.003	1.003
	ε =1.0%	1.008	1.007	1.009	1.009	1.008	1.009
	ε =1.5%	1.013	1.012	1.012	1.014	1.014	1.014
	ε =2.0%	1.016	1.016	1.019	1.018	1.019	1.019
Quan [12]		1.044	1.047	1.051	1.054	1.052	1.071

표 3 합성한 작업들(Class 2)에 대한 실험 결과

		Normalized Energy					
Jobs sets		$\mathcal{J}_1$	$\mathcal{J}_2$	$\mathcal{J}_3$	$\mathcal{J}_4$	$\mathcal{J}_5$	$\mathcal{J}_6$
#jobs		50	100	200	400	800	1600
FPTAS	ε =0.1%	1	1	1	1	1	1
	ε =0.5%	1.004	1.004	1.003	1.004	1.003	1.004
	ε =1.0%	1.009	1.007	1.007	1.008	1.009	1.009
	ε =1.5%	1.013	1.012	1.014	1.014	1.013	1.014
	ε =2.0%	1.018	1.016	1.018	1.018	1.019	1.019
Quan [12]		1.055	1.062	1.070	1.079	1.079	1.127

표 4 합성한 작업들(Class 3)에 대한 실험 결과

		Normalized Energy					
Jobs sets		$\mathcal{J}_1$	$\mathcal{J}_2$	$\mathcal{J}_3$	$\mathcal{J}_4$	$\mathcal{J}_5$	$\mathcal{J}_6$
#jobs		50	100	200	400	800	1600
FPTAS	ε =0.1%	1	1	1	1	1	1
	ε =0.5%	1.004	1.004	1.004	1.003	1.004	1.004
	ε =1.0%	1.009	1.007	1.007	1.009	1.008	1.009
	ε =1.5%	1.014	1.013	1.014	1.013	1.014	1.014
	ε =2.0%	1.018	1.017	1.019	1.018	1.019	1.019
Quan [12]		1.094	1.114	1.121	1.134	1.142	1.137

7. 결 론

본 연구에서는 가변 전압 프로세서로 구현된 고정 우선순위 실시간 시스템에 대한 최적의 전압 스케쥴링 문제를 고려하였다. 먼저, 이 문제가 NP-hard임을 증명하였다. 문제의 계산 복잡도를 고려할 때 이론적, 실용적 관점 모두에 대해 최선책인 fully polynomial time approximation scheme(FPTAS)를 제시하였다. 제시한 FPTAS는 임의의 $\varepsilon>0$에 대해 에너지 소모량이 최적해에 비해 $(1+\varepsilon)$배 이내에 드는 근사해를 작업들의 숫자와 $1/\varepsilon$의 다항함수로 표현되는 시간내에 계산한다. 실험결과, 제안한 FPTAS는 0.5%정도의 매우 적은 오차한계치에 대해서도 충분히 빠르게 작동하였다.

제안하는 FPTAS는 충분히 효과적이지만 여러 측면에서 확장할 수 있다. 예를 들어, 취할 수 있는 전압이 유한개로 제한되고 전압 전환시 오버헤드가 존재하는

보다 현실적인 프로세서 모델에 대해 적용할 수 있도록 FPTAS를 확장할 수 있을 것이다. 그 외에도, 온라인 알고리즘과의 상호작용을 고려한 오프라인 FPTAS를 고려할 예정이다.

참 고 문 헌

[1] T. Sakurai and A. Newton. Alpha-power Law MOSFET Model and Its Application to CMOS Inverter Delay and Other Formulars. IEEE Journal of Solid State Circuits, vol. 25, no. 2, pp. 584-594, 1990.

[2] Intel Corporation. Intel XScale Technology. http://developer.intel.com/design/intelxscale, 2001.

[3] AMD Corporation. PowerNow! Technology. http://www.amd.com, 2000.

[4] Transmeta Corporation. Crusoe Processor, http://www.transmeta.com, 2000.

[5] H. Aydin, R. Melhem, D. Mosse and P. M. Alvarez. Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems. In Proc. of Real-Time Systems Symposium, 2001.

[6] I. Hong, G. Qu, M. Potkonjak and M. B. Srivastava. Synthesis Techniques for Low-Power Hard Real-Time Systems on Variable Voltage Processors. In Proc. of Real-Time Systems Symposium, pp. 178-187, 1998.

[7] W. Kim, J. Kim and S. L. Min. A Dynamic Voltage Scaling Algorithm for Dynamic-Priority Hard Real-Time Systems Using Slack Time Analysis. In Proc. of Design, Automation and Test in Europe, 2002.

[8] P. Pillai and K. G. Shin. Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems. In Proc. of ACM Symposium on Operating Systems Principles, 2001.

[9] F. Yao, A. Demers and S. Shenker. A Scheduling Model for Reduced CPU Energy. In Proc. of IEEE Annual Foundations of Computer Science, pp. 374-382, 1995.

[10] F. Gruian. Hard Real-Time Scheduling for Low-Energy Using Stochastic Data and DVS Processors. In Proc. of International Symposium on Low Power Electronics and Design, pp. 46-51, 2001.

[11] Y. Shin and K. Choi. Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems. In Proc. of Design Automatioin Conference, pp. 134-139, 1999.

[12] G. Quan and X. Hu. Energy Efficient Fixed-Priority Scheduling for Real-Time Systems on Variable Voltage Processors. In Proc. of Design Automatioin Conference, pp. 828-833, 2001.

[13] G. Quan and X. Hu. An Optimal Voltage Schedule for Real-Time Systems on a Variable Voltage Processor. In Proc. of Design, Automation and Test in Europe, 2002.

[14] Y. Shin, K. Choi and T. Sakurai. Power Optimization of Real-Time Embedded Systems on Variable Speed Processors. In Proc. of International Conference on Computer-Aided Design, pp. 365-368, 2000.

[15] G. J. Woeginger. When Does a Dynamic Programming Formulation Guarantee the Existence of an FPTAS? In Proc. of ACM-SIAM Symposium on Discrete Algorithms, pp. 820-829, 1999.

[16] On Energy-Optimal Off-Line Scheduling for Fixed-Priority Hard Real-Time Systems On a Variable Speed Processor. Technical report, 2003. Available from http://davinci.snu.ac.kr/Download/opt_fp_vs.pdf.

[17] A. Ben-Tal and A. Nemirovski. Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications. SIAM, 2001.

[18] M. Garey and D. Johnson. Computers and Intractability. W.H. Freeman and Company, 1979.

[19] N. Kim, M. Ryu, S. Hong, M. Saksena, C. Choi, H. Shin. Visual Assessment of a Real-Time System Design: A Case Study on a CNC Controller. In Proc. of Real-Time Systems Symposium, pp. 300-310, 1996.

[20] C. Locke, D. Vogel and T. Mesler. Building a Predictable Avionics Platform in Ada: A Case Study. In Proc. of Real-Time Systems Symposium, 1991.

윤 한 샘

1998년 서울대학교 전기공학부 학사. 2000년 서울대학교 전기공학부 석사. 2001년~현재 서울대학교 컴퓨터공학부 박사과정 재학중. 관심분야는 실시간 시스템 및 내장형 시스템임

김 지 홍

정보과학회논문지 : 시스템 및 이론 제 31 권 제 1 호 참조