실시간 시스템 설계를 위한 주기 할당 알고리즘

(A Period Assignment Algorithm for Real-Time System Design)

유 민 수 † 홍 성 수 ††
(Minsoo Ryu) (Seongsoo Hong)

요 약 산업용 실시간 시스템에서 사용되는 디지털 제어기는 상호 작용하는 주기적 태스크들로 이루어져 있다. 이러한 태스크들은 최대 수행 주기(maximum activation periods)를 제약조건으로 가짐으로써, 요구되는 제어특성을 유지한다. 그러므로 실시간 시스템을 개발하는데 있어 필수적인 단계는 각각의 태스크에게 자원 이용률을 최소화하면서 최대 수행 조건을 만족시킬 수 있는 고정된 주기를 할당하는 것이다[1].

각각의 태스크간의 생산자/소비자(producer/consumer)관계를 나타내주는 태스크 그래프와 자원 요구량, 그리고 주기에 대한 범위제한이 주어진다면 주기 할당 문제는 비선형 최적화 문제가 된다. 이 논문에서는 최적해의 자원이용률의 두 배를 넘지 않는 해를 찾을 수 있는 선형 수행시간의 근사 알고리즘을 제시한다. 실험 결과에서 알 수 있듯이, 제안된 알고리즘은 대부분의 경우 최적해에 매우 근접한 해를 가지게 된다.

Abstract Digital controllers found in many industrial real-time systems consist of a number of interacting periodic tasks. To sustain the required control quality, these tasks possess the maximum activation periods as performance constraints. An essential step in developing a real-time system is thus to assign each of these tasks a constant period such that the maximum activation requirements are met while the system utilization is minimized [1].

Given a task graph design allowing producer/consumer relationships among tasks [2], resource demands of tasks, and range constraints on periods, the period assignment problem falls into a class of nonlinear optimization problems. This paper proposes a ploynomial time approximation algorithm which produces a solution whose utilization does not exceed twice the optimal utilization. Our experimental analysis shows that the proposed algorithm finds solutions which are very close to the optimal ones in most cases of practical interest.

1. 서 론

실시간 시스템은 대체로 상호 작용하는 태스크들로 이루어진다. 이러한 태스크중 대부분은 고정된 주기에 따라 생산자 태스크로부터 데이타를 읽고 소비자 태스크에 데이타를 쓴다. 피드백(feedback) 제어 시스템이 그 전형적인 예이다. 제어 태스크는 주기적으로 입력데이타를 읽고, 제어연산을 하며, 구동기에 필요한 출력을 만들어내는데, 이러한 모든 작업들은 주기적으로 이루어져야 한다.

실시간 태스크들은 요구되는 제어특성을 보장하기 위해서, 주어진 작업명세에 의해 결정되는 최대 수행 주기를 시간 제약으로 가지게 된다. 그러므로 실시간 시스템을 개발하는데 있어 필수적인 단계는 각각의 태스크에게 자원이용률을 최소화하면서 최대 수행 조건을 만족시킬 수 있는 고정된 주기를 할당하는 것이다. 이때 중

· 본 논문은 국방과학 연구소 및 서울대학교 자동제어특화센터의 연구비 지원, 한국과학재단의 핵심전문연구(981-0924-127-2)의 연구비 지원, 과학기술부의 국가지정연구실 연구비 지원하에 연구되었음.
† 비 회 원 : 서울대학교 전기공학부
msryu@redwood.snu.ac.kr
†† 종신회원 : 서울대학교 전기공학부 교수
sshong@redwood.snu.ac.kr
논문접수 : 1998년 2월 22일
심사완료 : 1999년 10월 15일

간 태스크들이 다른 제약을 가지는 태스크들에 의해서 공유된다면, 이 문제는 비선형 최적화 문제가 된다.

Gerber 등은 이 문제를 실시간 시스템 설계 문제의 영역으로 접근하여, 그들의 설계 방법론에서 비선형 최적화 문제로 정의하였다[1]. Gerber 등은 시스템의 입력과 출력에 대한 양극단 시간 제약(end-to-end timing constraint)을 정의하여 시스템의 시간적 특성을 기술하고, 태스크간의 데이타 흐름을 나타내는 태스크 그래프로서 시스템의 가능적 특성을 기술하였다. 이 때 주어진 양극단 시간 제약을 충족하도록 태스크들의 주기 및 종료시한을 결정하는 것이 설계 이론의 핵심으로서, 이를 위한 휴리스틱이 제시되었다. 제안된 알고리즘은 주로 탐색 공간(search space)을 줄이는데 치중하였지만, 적정한 크기의 문제에 대해서도 매우 큰 탐색공간을 가지기 때문에 실제적으로 적용되기에는 문제가 있다. 지금까지 알려진 바로는 이러한 주기 할당 문제를 위해, 알려진 동작특성을 갖는 선형 수행시간 근사 알고리즘은 제안된 적이 없었다.

Seto 등도 실시간 제어 시스템에서의 유사한 문제에 대하여 접근하였다[3]. 제어 수행 요구 조건과 스케줄링 가능성 제약이 샘플링 주기의 범위를 결정한다고 하면, 그들이 제시한 알고리즘은 제어 시스템의 성능 지수(performance index)를 최적화 하는 샘플링 주기를 결정한다. 그러나 [3]에서의 어플리케이션 모델은 매우 단순하다. 예를들어 [3]에서는 시스템의 모든 태스크가 독립적으로 동작하는 반면 [1]에서는 태스크 그래프에서 태스크들간의 생산자/소비자 관계 설정 및 데이타 공유가 가능하다.

본 논문에서는 [1]에서 제기된 실시간 설계 문제를 위한 선형 수행시간 주기 할당 알고리즘을 제시한다. 이 알고리즘은 (1) 최적 최대공약수 할당법과 (2) 출력 태스크간의 조화 주기 할당이라는 두가지 효과적 휴리스틱을 이용한다. 첫 번째 휴리스틱에서는 생산자 태스크의 주기는 소비자 태스크 주기의 최대공약수가 된다. 두 번째에서는 소비자 태스크를 가지지 않는 출력 태스크들은 조화 주기를 가지게 된다. 논문에서는, 제시된 알고리즘이 최악의 경우에 대해 최적 자원이용률의 두 배를 넘지 않는 자원이용률을 가지는 해를 이끌어낸다는 것을 증명한다. 실험에 의하면, 실제로는 대부분의 경우 최적해와 매우 가까운 해를 구하게 된다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 실시간 시스템 설계의 태스크 그래프를 정의함으로써, 여러 제약들 및 목적함수와 함께, 주기 할당 문제에 대한 정의를 내린다. 3장에서는 두 가지 휴리스틱과 근사 알고리즘을 제시한다. 4장에서는 제시된 알고리즘에 대한 분석과 최악의 경우에 대한 성능상한(performance bound)를 증명한다. 5장에서는 완전(exhaustive) 탐색을 사용하는 최적 알고리즘과 제시된 알고리즘을 실험을 통해 비교함으로써, 제시된 알고리즘을 평가한다. 6장에서는 본 논문에서 논의된 내용을 정리한다.

2. 문제 정의

주기할당 문제에 있어서 실시간 시스템은 순환하지 않으며 방향성을 가지는 태스크 그래프 $G(V,E)$로 표현되는데, V는 태스크들의 집합 $\{p_1, p_2, \ldots, p_n\}$이며, E는 태스크간의 연결관계를 나타내는 화살표들의 집합이다. 각각의 태스크는 다양한 방법으로 측정 가능한 제한된 수행시간 e_i를 가진다[4]. $p_i \rightarrow p_j$는 생산자/소비자 관계를 나타내는 것으로서, p_i가 생산한 데이타를 p_j가 소비한다는 의미를 가진다[2].

T_i를 p_i의 주기라고 하면, 각각의 생산자/소비자 관계 $\langle p_i, p_j \rangle$에서 T_j는 T_i의 정수배가 되는 하모닉주기(harmonic period)를 가져야 하며, "$T_i \mid T_j$"로 표현된다. 그림 1은 하모닉주기가 할당된 태스크 그래프의 예이다. 만일 태스크들이 임의의 주기로 수행된다면, 태스크들의 수행은 위상이 어긋나게되어 통신상의 지연(latency)이 커지게 된다. 하모닉주기는 두 태스크가 위상일치를 이룸으로써 통신 지연을 줄일 수 있게 해준다[1]. 또한 공통 생산자로부터 여러 소비자로의 예측 가능한 데이타 흐름을 보장해 준다. 예를 들어, 그림 1에서 p_4는 p_3가 생산하는 모든 데이타를 받고, p_5는 p_3가 생산하는 데이타중 매 세 번째 것들을 받는 것이 보장된다. 본 논문에서는 아래와 같은 태스크 집합을 사용한다.

P_{tail}	외선(outgoing edge)이 존재하지 않는 태스크들의 집합. (예: 그림 1에서 p_4 , p_5 , p_6)
P^i_{tail}	p_i에서 접근가능한 출력 태스크 p_j들의 집합 (예: 그림 1에서 $P^3_{tail} = \{ p_4 , p_5 \}$)
P^i_{succ}	p_i로부터의 내선(incoming edge)을 가지고 있는 태스크들의 집합

모든 출력 태스크 $p_i \in P_{tail}$은 T_i에 대해서 $1 \le T_i \le T_i$"인 제약을 가지는데, 이때 T_i"는 수행요구조건에 의해 결정되는 최대 주기 제약이다. 주기 조화성 제약 때문에 모든 비출력 태스크 p_j 또한 $1 \le T_j \le T_i$"인

주기 제약을 가지는데, 이때 T_i^u는 모든 소비자 태스크들의 최대 주기중 가장 작은 값이다. 그러므로 비출력 태스크 p_i의 T_i^u는 $T_i^u = \min\{ T_j^u \mid p_j \in P^i_{succ}\} = \min\{ T_j^u \mid p_j \in P^i_{tail}\}$으로 결정된다.

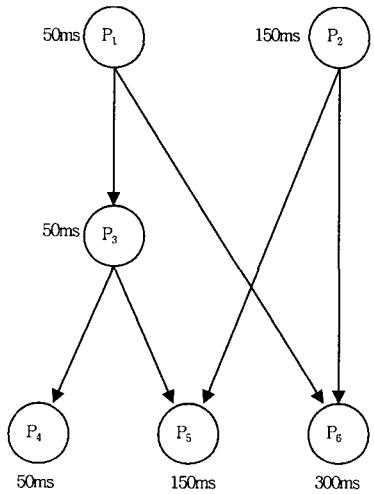


그림 1 태스크 그래프

주기할당 문제에서 목적함수(objective function)는 시스템의 스케줄링 가능성을 최대화하는 것이다. [1]에서와 같이 목적함수로서 자원이용률 $U = \sum^n_{i=1} e_i / T_i$의 최소화를 채택한다. 비선점형, 일정표 기반(calendar-based) 스케줄링[5,6], 선점형, 정적, 동적 우선순위 스케줄링[7] 등의 실시간 스케줄링 알고리즘에 따라서 수많은 스케줄링 가능성의 척도가 존재한다. 본 논문에서는, 선점형 우선순위 스케줄링 방법을 가정한다. RMS나 EDF와 같은 선점형 우선순위 스케줄링에서 자원이용률은 실시간 시스템의 스케줄링 가능성에 대한 충분한(sufficient) 측정도구로 널리 알려져 있다[7, 8].

결과적으로, 주어진 문제는 다음과 같다.

"태스크 그래프와 태스크 주기에 대한 범위 제약이 주어졌을 때, 각각의 태스크에게, 범위 제약을 만족하고 시스템의 자원이용율 $U = \sum^n_{i=1} e_i / T_i$ 를 최소화하는 조화 주기를 할당한다."

3. 주기 할당 알고리즘

주기조화성 제약과 목적함수 때문에, 본 문제는 비선형 최적화 문제가 되며 짧은 시간 안에 해를 찾기 위한 방법으로, 자원이용률을 최소화하는 두 가지의 휴리스틱으로 이루어진 근사 알고리즘을 제안한다.

3.1 최적 최대공약수(GCD) 할당법

첫 번째 휴리스틱은 최적 최대공약수 할당법이다. 이것은 $T_i = GCD\{ T_j \mid p_j \in P^i_{succ}\}$인 T_i를 비출력 태스크인 p_i에 할당하는 역방향 주기 할당 방법이다. 이러한 주기 할당법은 출력 태스크의 바로 위에 존재하는 비출력 태스크로부터 시작하며, 모든 비출력 태스크에 주기가 할당될 때까지 반복적으로 적용된다. 최적 최대공약수 할당법의 결과로서, 생산자 태스크는 항상 소비자 태스크 주기의 조화 주기중 가장 큰 값을 가지게 된다.

정리 1은 최적 최대공약수 할당법이 전체 시스템에 최적 주기를 할당하는 필요조건임을 증명한다.

정리 1. 출력 태스크들의 주기가 모두 최적 주기값으로 주어지면, 최대공약수 할당법은 나머지 태스크들에게 최적 주기값을 할당한다.

증명. 어떤 최적해가 최대공약수 주기값이 아닌 T_i를 가지는 중간 태스크 p_i를 가진다고 가정해 보자. $T_i^{GCD} = GCD\{ T_j \mid p_j \in P^i_{succ}\}$라 하면, 주기 조화성에 의해 T_i는 P^i_{succ}에 속한 태스크들이 취하는 주기값의 공약수가 되므로 $T_i \mid T_i^{GCD}$ 이다. 그러므로, T_i는 다른 태스크의 주기값에 영향을 주지않고 T_i^{GCD}값으로 바뀔 수 있으며, 더 낮은 자원이용률을 초래하게 된다. 이것은 가정에 위배되므로 최적해는 최대공약수 할당법에 의해 얻어진다 (Q.E.D).

정리 1은 최적 주기값 할당에서 출력 태스크에만 초점을 맞출 수 있게 해주기 때문에 문제의 크기를 줄여주는 효과가 있다.

3.2 출력 태스크의 조화 주기 할당

두 번째 휴리스틱은 출력 태스크에 대한 조화 주기 할당법이다. 자원이용률을 최소화 하기 위해서는, 비출력 태스크에 큰 공약수 값이 할당될 수 있도록 출력 태스크에 가능한 가장 큰 조화 주기값을 할당하는 것이 바람직하다. 그러기 위해서, 본 알고리즘은 출력 태스크간에 추가적인 주기 조화성 제약을 도입한다. 이 제약을 만족시키기 위해서 출력 태스크에 조화 주기값을 할당한다. 이 방법은 다음과 같은 절차에 의해 이루어 진다. 첫 번째로 출력 태스크들을 T_i^u가 감소하지 않는 순서대로 정렬한다. 두 번째로 정렬된 태스크중 인접한 두 개의 태스크들 사이에 주기조화성을 성립시킨다. 이러한 부가적인 주기조화성은 주어진 태스크 그래프를 약간만 수정하여 나타낼 수 있다. 그림 2는 그림 1에 추가된 주기조화성 제약이 적용된 후의 태스크 그래프이다.

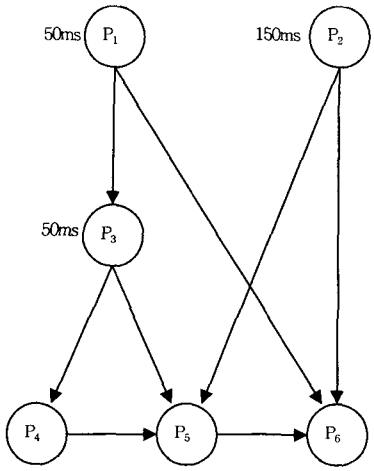


그림 2 출력 태스크에 주기 조화성 제약이 추가되어 수정된 태스크 그래프

출력 태스크들간의 주기조화성 제약 때문에 원래의 주기할당 문제는 출력 태스크에 최소의 T^u를 할당하는 문제가 된다.

3.3 알고리즘

전체 알고리즘은 아래와 같다. (1)단계에서는 출력 태스크를 최대 주기 제약값이 감소하지 않는 순서대로 정렬한다. (2)단계에서는 최대 주기 제약값중 최소값으로 출력 태스크의 주기값을 결정하는데, 최악의 수행시간 결과가 한계를 가지게 하기 위해서 $\lceil T_1^u/2 \rceil$ 와 T_1^u사이 값을 가지는 T_1을 선택한다. 이렇게 해야하는 이유는 다음 장에서 증명할 것이다. (3)단계에서는, 추가된 주기 조화성에 따라 다른 출력 태스크에 최대 조화 주기값을 할당한다. 끝으로 (4)단계에서는 최대공약수 할당법에 의해 비출력 태스크에 주기값을 할당한다.

다음은 제안된 주기할당 알고리즘을 기술한 것이다.

주기 할당 알고리즘

{

(1) $T_1^u \le T_2^u \le \ldots \le T_m^u$을 만족하도록 P_{tail}을 정렬한다.
→ $\langle p_1, p_2, \ldots, p_m \rangle$

(2) $\lceil T_1^u/2 \rceil \le T_1 \le T_1^u$ 를 만족하는 임의의 T_1을 선택한다.

(3) $T_i = \lfloor \frac{T_i^u}{T_{i-1}} \rfloor \cdot T_{i-1}$, for $2 \le i \le m$

(4) 비출력 태스크에 대해 최대공약수 할당 수행

}

n개의 정렬된 태스크들을 가지는 태스크 그래프에서 (출력 태스크들은 T_i^u가 감소하지 않게 정렬되고, 비출력 태스크들은 위상적으로(topologically) 정렬된다), 제안된 알고리즘은 O(n)의 주기 할당시간이 요구된다. 알고리즘의 최대공약수 할당은 식(3)에 의해서 간단하게 계산될 수 있다. 최대공약수 할당에서 $GCD(T_i, \ldots, T_j) = T_i$ 이 이용될 수 있다.

4. 알고리즘 분석

본 절에서는 제시된 알고리즘을 최악의 경우에 대해 분석하였으며, 그 결과로 정리 2에 요약하였다. 이 알고리즘을 이용하면 최악의 경우에도 자원이용률이 최적 자원이용률의 두 배가 넘지 않는 해를 구할 수 있다.

정리 2. U_{alg}를 본 알고리즘에 의해 계산된 자원이용률, U_{opt}를 최적 자원이용률이라 하면, U_{alg}는 항상 $2U_{opt}$보다 작다.

증명. $\langle p_1, p_2, \ldots, p_m \rangle$을 P_{tail}의 정렬된 리스트라고 하자. $(T_1, \ldots, T_m)$을 출력 태스크에 할당된 주기값이라고 가정하자. 정리 1에 의하여, V에 속하는 비출력 태스크의 주기값은 $\{T_1, \ldots, T_m\}$의 조합으로 표현될 수 있다. 따라서 일반적으로 자원이용률 U는 아래와 같이 나타낼 수 있다.

$$U = \frac{\hat{e}_1}{T_1} + \frac{\hat{e}_2}{T_2} + \ldots + \frac{\hat{e}_m}{T_m} + \frac{\hat{e}_{m+1}}{GCD(T_1, T_2)} + \frac{\hat{e}_{m+2}}{GCD(T_2, T_3)} + \ldots + \frac{\hat{e}_{2^n-1}}{GCD(T_1, T_2, \ldots, T_m)} \quad (1)$$

이때, $\hat{e}_i = \sum_{p_j \in Q_i} e_j$ 이며 Q_i는 i번째 분모에 해당하는 출력 태스크로의 경로를 가지는 태스크 p_j들의 집합이다. 따라서, Q_i가 공집합이면, $\hat{e}_i = 0$가 된다.

알고리즘의 성능 분석을 위해 U의 하한값인 U_{low}를 (1)에서 유도한다. 즉 (1)에서 각 항의 분모가 취할 수 있는 최대값들을 취하면 아래와 같다.

$$U_{low} = \frac{\hat{e}_1}{T_1^u} + \frac{\hat{e}_2}{T_2^u} + \ldots + \frac{\hat{e}_m}{T_m^u} + \frac{\hat{e}_{m+1}}{T_1^u} + \frac{\hat{e}_{m+2}}{T_2^u} + \ldots + \frac{\hat{e}_{2^n-1}}{T_1^u} \quad (2)$$

본 논문에서 제시된 알고리즘은 출력 태스크 P_{tail}의 주기조화성을 가정하기 때문에, $1 \le i < j \le m$을 만족하는 모든 i, j에 대해 $GCD(T_i, \ldots, T_j) = T_i$이 성립한다. 따라서 제시된 알고리즘에 의해 얻어진 해의 자원이용률 U_{alg}는 (1)로 부터 아래와 같이 유도될 수 있다.

$$U_{alg}=\frac{\hat{e}_1}{T_1}+\frac{\hat{e}_2}{T_2}+\ldots+\frac{\hat{e}_m}{T_m}+\frac{\hat{e}_{m+1}}{T_1}+\frac{\hat{e}_{m+2}}{T_2}+\ldots+\frac{\hat{e}_{2^m-1}}{T_1} \quad (3)$$

(2)와 (3)에서 각 항의 값을 비교하면 U_{alg}와 U_{low}의 관계식을 유도할 수 있다. u_i^{low}와 u_i^{alg}를 각각 U_{alg}와 U_{low}의 i번째 항이라 하자.

(a) $i=1$: 제시된 알고리즘에 의해 $T_1 \geq \lceil T_1^u/2 \rceil$ 이므로, $u_i^{alg} \leq 2\ u_i^{low}$

(b) $2 \leq i \leq 2^m-1$, $u_i^{low} \neq 0$: $1<j \leq m$일 때,

$$\frac{u_i^{alg}}{u_i^{low}}=\frac{T_j^u}{\lfloor T_j^u/T_{j-1} \rfloor \cdot T_{j-1}}$$

$x \geq 1$일때 $\frac{x}{\lfloor x \rfloor}<2$ 이므로, $u_i^{alg}<2\ u_i^{low}$이 성립한다.

위의 결과들에 의해 $U_{alg}<2\ U_{low}$이 성립하며, U_{opt}를 최적 자원이용률이라 하면, 정의에 따라 이는 자원이용률의 하한인 U_{low}보다 클 수 밖에 없으므로 $U_{low} \leq U_{opt}$로 나타낼 수 있다. 이를 정리하면, 다음의 결과를 얻을 수 있다.

$$U_{alg} < 2U_{opt}$$

(Q.E.D)

5. 실험 및 결과

앞의 논의에서 증명된 것처럼, 제안된 알고리즘과 최적 알고리즘간의 성능비(performance ratio)는 2.0보다 작았다. 그러나 이 결과는 최악의 경우에 대한 분석으로부터 얻어진 것이다.

이 절에서는 제안된 알고리즘의 성능이 실제적인 경우에는 최적화 알고리즘에 매우 가깝다는 것을 실험을 통하여 알아보도록 하겠다.

비교 실험을 위해서, 제시된 알고리즘과 완전(exhaustive) 탐색에 의한 최적 알고리즘을 구현하였고, 제어 시스템의 알고리즘 구조에서 자주 접하게 되는 다섯가지의 대표적 테스크 그래프를 이용하여 인위적인 부하(workload)를 생성하였다. 여기서 사용된 태스크 그래프는 인-트리(In-Tree), 아웃-트리(Out-Tree), 포크-조인(Fork-Join), 라플라스 방정식 풀이법(Laplace Equation Solver)[9], FFT(Fast Fourier Transform)[10] 형태이다. 그림 3은 이들 태스크 그래프를 보여주고 있다. 각각의 그래프 형태에 대하여, 태스크의 갯수와 최대 주기 제약을 변화시켜서 서로 다른 세 가지의 태스크 그래프를 만들었다. 한 개의 출력 태스크만을 가지는 인-트리, 포크-조인, 라플라스 방정식 풀이법 태스크 그래프에서, 단순하게 주기할당이 이루어지는 것을 막을 수 있도록 이들 그래프에 두 개이상의 출력 태스크를 더 첨가하였다. 한편, 완전 탐색 알고리즘이 갖는 시간 복잡도 때문에, 10~20개의 태스크를 가지는 작은 크기의 문제에 대해서 실험을 수행하였다.

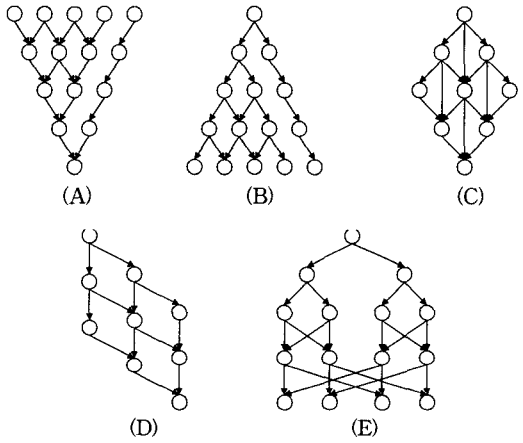


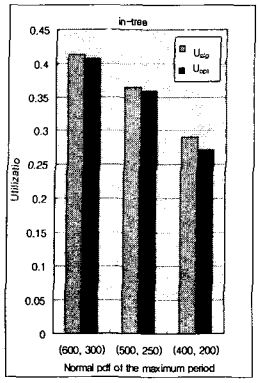
그림 3 태스크 그래프 형태: (A) 인-트리, (B) 아웃-트리, (C) 포크-조인, (D) 라플라스 방정식 풀이법, (E) FFT

각각의 태스크 그래프 형태에 대해서 태스크 수행시간과 최대 주기 제약을 변화시킴으로써 부하를 생성하였다. 출력 태스크의 최대 주기 제약에는 평균이 μ이고 표준편차가 σ인 정규확률분포 $N(\mu, \sigma^2)$를 사용하였다. 구체적으로 각각의 태스크 그래프 형태에 대해서 $N(600, 300^2)$, $N(500, 250^2)$, $N(400, 200^2)$인 세가지 경우에 대해서 실험을 하였다. 태스크 수행시간 e_i는 모든 실험에서 $N(10, 5^2)$이 사용되었다.

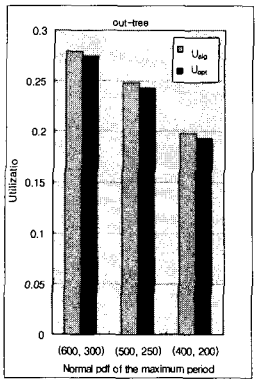
그림 4에 실험결과가 요약되어 있다. 그림 4에서 볼 수 있듯이 제안된 알고리즘은 최적의 경우와 매우 가까운 해를 구한다. 본 실험에서 평균적인 성능비는 $\frac{U_{alg}}{U_{opt}}=1.0330$ 이었다.

6. 결 론

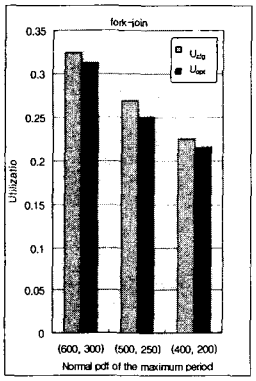
본 논문에서는 O(n)의 주기 할당 단계를 가지며 최적해에 근접한 해를 구할 수 있는 주기할당 알고리즘을 제시하였다. 아울러 제시된 알고리즘이 $\frac{U_{alg}}{U_{opt}} \leq 2$ 인 성능비를 보임을 증명하였고, 실제로는 최적해와 매우 근접



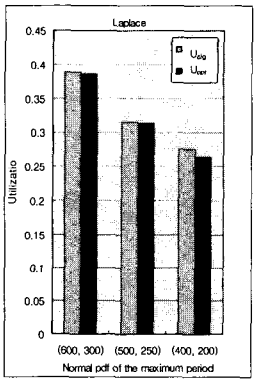
(A)



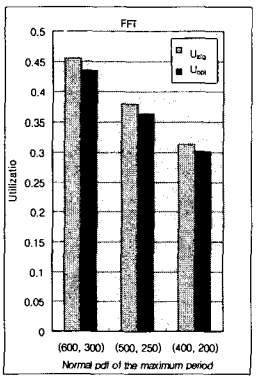
(B)



(C)



(D)



(E)

그림 4 성능 비교 : (A) 인-트리, (B) 아웃-트리, (C) 포크-조인, (D) 라플라스 방정식 풀이법, (E) FFT

한 해를 가진다는 것을 실험으로 입증하였다. 실험에서는 평균적으로 $\frac{U_{alg}}{U_{opt}} = 1.0330$ 인 성능비를 보였다.

제안된 알고리즘은 [1]에서 제시된 실시간 시스템 설계 방법론의 필수적인 요소로 이용될 수 있다. 이 방법론은 현재 분산 플랫폼(*distributed platform*) 위에서의 실시간 시스템 설계에 적용될 수 있으며, 이를 위해 본 알고리즘을 발전시키고 있다.

참 고 문 헌

[1] R. Gerber, S. Hong, and M. Saksena, "Guaranteeing real-time requirements with resource-based calibration of periodic processes," *IEEE Transactions on Software Engineering*, 21(7):579-592, July 1995.

[2] K. Jeffay, "The real-time producer/consumer paradigm: A paradigm for the construction of efficient, predictable real-time systems" In *ACM/SIGAPP Symposium on Applied Computing*, pages 796-804. ACM Press, February 1993.

[3] D. Seto, J.P. Lehoczky, L. Sha, and K.G. Shin, "On task schedulability in real-time control systems" In *Proceedings of IEEE Real-Time Systems Symposium*, pages 13-21, December 1996.

[4] C. Park and A. Shaw, "Experimenting with a program timing tool based on source-level timing schema" In *Proceedings of IEEE Real-Time Systems Symposium*, pages 72-81, December 1990.

[5] J. Xu and D. Parnas, "Scheduling processes with release times, deadlines, precedence and exclusion relations" *IEEE Transactions on Software Engineering*, 16(3):360-369, March 1990.

[6] T.F. Abdelzaher, E.M. Atkins, and K.G. Shin, "QoS negotiation in real-time systems and its application to automated flight contro" In *Proceedings of IEEE Real-Time Technology and Applications*, pages 228-238,December 1997.

[7] C. Liu and J. Layland. "Scheduling algorithms for multiprogramming in a hard real-time environment" *Journal of the ACM*, 20(1):46-61, January 1973.

[8] J. Lehoczky, L. Sha and Y.Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior" In *Proceedings of IEEE Real-Time Systems Symposium*, pages 166-171, December 1989.

[9] M.Y. Wu and D.D. Gajski. "Hypertool: A programming aid for message-passing systems" *IEEE Transactions on Parallel and Distributed Systems*, 1(3):330-343, July 1990.

[10] V.A.F. Almeida, I.M. Vasconcelos, J.N.C. Arabe, and D.A. Menasce, "Using random task graphs to

investigate the potential benefits of heterogeneity in parallel systems" In *Proceedings of Supercomputing*, pages 683-691, 1992.

[11] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, "A resource allocation model for QoS management" In *Proceedings of IEEE Real-Time Systems Symposium*, pages 298-307, December 1997.

유 민 수

1995년 서울대학교 제어계측공학과 학사. 1997년 서울대학교 전기공학부 석사. 1997년 3월 ~ 현재 서울대학교 전기공학부 박사과정. 관심분야는 실시간 시스템 설계, 운영체계, 실시간 통신, 실시간 제어 시스템, 분산 시스템

홍 성 수

1986년 서울대학교 컴퓨터공학과 졸업. 1988년 서울대학교 컴퓨터공학과 석사학위 취득. 1994년 University of Maryland at College Park, Dept. of Computer Science 박사. 1988년 2월 ~ 1989년 7월 한국전자통신연구원 연구원. 1994년 12월 ~ 1995년 3월 Faculty Research Associate(University of Maryland at College Park). 1995년 4월 ~ 1995년 8월 Silicon Graphics Inc. 연구원. 1995년 9월 ~ 1997년 9월 서울대학교 전기공학부 전임강사. 현재 동학부 조교수. 관심분야는 실시간 시스템, 실시간 운영체제, 분산제어시스템, 실시간 시스템 설계 방법론, 멀티미디어 시스템, 소프트웨어 공학