'90 韓國自動制御學術會議論文集 1990. 10.26~27

다중 로보트 시스템의 작업 스케쥴링 및 성능 평가

이 기동 조 혜경 이 범희 고 명삼

로보틱스 및 지능 시스템 연구실
서울대학 대학원 제어 계측 공학과

Job Scheduling and Performance Evaluation of MRS(Multi Robot System)

Ki-Dong Lee Hye-Kyung Cho Bum-Hee Lee Myoung-Sam Ko

Robotics and Intelligent Systems Laboratory
Dept. of Control and Instrumentation Engineering
Seoul National University

Abstract

A mixed IP formulation is presented which concurrently routes and sequences the tasks on the MRS, reflecting the flexible characteristics. As a preliminary work for the performance evaluation of the MRS, a two robot system working on independent tasks is studied. Models for three types of the system capable of simple error recovery function are established using queueing model, and performances are evaluated and analysed.

1. 서론

전통적인 일정계획 (scheduling) 은 부품의 이동 경로를 결정하는 문제 (routing) 와 이들을 처리하는 순서를 결정하는 문제 (sequencing) 를 연속적인 두 단계로 나누어 고려해 왔다 [1]. 이와 같은 접근 방법은 하나의 기계의 능력이 한 가지로 제한되는 기존의 기계들을 대상으로 하고 있으므로, 로보트의 경우에는 그 특징인 다양성 (versatility) 과 유연성 (flexibility) 을 전혀 고려하지 않은 결과를 얻게 된다. 본 논문에서는 MRS(Multiple Robot System)에 적용하기 위한 일정계획 문제의 특징을 알아보고, 이를 반영하여 전술한 두 단계의 과정을 동시에 고려하는 일정계획 문제[2]를 혼합정수계획법 (mixed Integer Programming)으로 정형화 한다.

또한 MRS의 성능 평가를 위해 MRS를 일종의 MPS(Multiple Processor System)으로 볼 수 있다. MPS에는 여러가지 구조가 있으나 크게 두가지로 나눌 수 있다. 첫째는 느슨히 결합된(loosely coupled)MPS이고 다른것은 긴밀히 결합된 (tightly coupled)MPS이다. 전자는 전송되는 데이타를 기본으로(message based) 분석하며 성능 평가 지표(Performance Index)로는 프로세서간의 최대 거리, 프로세서간의 평균 거리, 프로세서의 사용 빈도 등이 있을 수 있다. 후자는 공유하는 메모리 또는 버스의 상태를 분석하며 여기에는 다시 공통 버스(common bus)방식[3], 십자형 교환(crossbar switch)방식[4], 다중 버스/다중 메모리(multibus/multiport memory)방식[5] 등이 있고 그것의 성능 평가 지표로는 생산률(throughput), 반응 시간(response time) 사용률(utilization) 등이 있다[6].

성능 평가를 위해 채택할 수 있는 모델은 여러가지가 있으며 목적 또한 다양하지만 주로 큐잉 모델, 페트리 망 (Petri Net), 마코프 체인(Markov Chain) 등이 많이 사용된다.

2. 연구 배경

2.1 로보트 일정계획의 특징

일정계획 문제 자체가 내포한 NP hard 특성으로 인하여 일반적인 큰 규모의 문제에까지 적용할 수 있는 최적해 (optimal solution) 를 찾기란 불가능하다. 따라서 일정계획에 관한 연구들은 작은 규모의 문제에 대한 최적해를 다루거나, 간단한 형태의 해가 구해지도록 문제를 축소시킨 것이 대부분이며, 경험적인 지식을 이용하는 방법, 실시간 처리에 중점을 두는 방법 [7] 등 준 최적해 (suboptimal solution) 를 빠른 시간에 구하는 문제에도 관심이 모아지고 있다.

유연성과 다양성을 고려한 로보트 일정계획은 순수한 경로 결정문제 (pure routing) 나, 순수한 순서 계획 문제 (pure sequencing) 만을 다루어 온 전형적인 생산 일정계획과 달리 다음과 같은 특징을 고려해야 한다.

- 기계가 매우 유연한 경우이므로, 경로 결정 및 순서 결정 문제를 동시에 고려해야 한다.
- 두 대의 기계가 동시에 작업할 때, 충돌 문제를 고려해야 한다.
- 도구 (tool) 교환 시간에 대한 분석이 필요하다.
- 작업 진행에 영향을 미치는 센서가 있는 경우 이들도 하나의 기계로 보고 분석할 필요가 있다.
- 두 로보트의 협동작업의 경우 두 기계의 동기를 위해 불가피한 여유 시간이 발생하므로 이를 효과적으로 처리해야 한다.

2.2 MRS 구조의 특징 및 성능 평가를 위한 가정

MRS는 프로세서가 미리 지정된 작업을 순차적으로 행하기 때문에 긴밀히 결합된MRS로 해석하는 것은 별 의미가 없으며 로보트가 동작시 오류가 생기는 것을 고려하기 위해서는 확률적 성질을 가진 모델을 채택해야 한다.

로보트가 수행하는 작업(task)은 다른 로보트와 관계없이 할 수 있는 독립 작업과 그렇지 않은 종속 작업이 있다. 종속 작업에는 다시 다른 로보트와 동시에 하는 공동 작업(coordinate task)과 작업 시간의 순서에 관계있는 순차 작업(time sequential task), 다른 로보트의 상태에 관계있는 제한 작업(constraint task) 으로 나눌 수 있다.

오류 회복을 위한 MRS을 분석하기 위해서 MRS를 느슨히 결합된 MPS로 간주하고 모델은 큐잉 모델을 사용한다. 그러면 로보트가 수행해야 할 명령들이 큐의 방문자(cutsomer)가 되며 행위자(server)는 로보트가 된다. 여기서 고려할 성능 평가 지표는 전체 실행 완료 시간과 실행 실패 확률, 실행 실패하여 남은 명령 갯수의 기대값 등이다. 작업의 종류는 독립 작업으로 하고 두대의 로보트가 사용될 경우를 대상으로 한다. 또한 시스템 해석을 위해 다음 가정을 한다. 1) 로보트 실행 시간의 분포(distribution)는 지수분포(exponential distribution)이다. 2) 명령을 전송하는 도중에는 오류가 발생하지 않는다. 3) 로보트에 오류가 발생했을 경우 그다음 부터는 그 로보트를 사용하지 못한다. 즉 오류 복구가 즉시 이루어지지 않는다.

3. MRS의 일정계획

3.1 문제의 설정

로보트 일정계획 문제를 특징 짓는 요소로는 여러 가지를 들 수 있다. 먼저 부품 입력에 대한 가정에 따라 일정계획 문제는 그 접근 방법이 크게 달라진다. 임의의 부품이 도착하는 것으로 가정하면 동적일정계획(dynamic scheduling) 문제가 되며, 모든 부품이 항상 이용가능한 것으로 가정하면 정적일정계획 (static scheduling) 문제를 풀게 된다. 이 밖에도 로보트의 수, 각 로보트의 기능, 작업의 형태, 작업 처리 시간, 도구의 사용 여부등에 따라 일정계획 문제의 복잡도가 달라지는데 [8], 본 논문에서는 다음과 같은 도구 교환 시간만을 제외한 매우 일반적인 경우를 다룬다.

- 정적 일정계획
- 로보트의 수 : K 개
- 로보트의 기능 : 서로 다름
- 작업 지정 형식 :
  - Rk (로보트 k 만이 처리 가능한 작업)
  - RloR2 (모든 로보트가 같은 능력을 갖는 작업)
  - RloR2p (모든 로보트가 처리할 수 있으나 로보트마다 처리 능력이 다른 작업)
  - RlaR2 (두 로보트의 공동 작업)

3.2 정수계획법을 이용한 일정계획

N개의 작업을 K대의 로보트에 분배하여 처리하는 순서를 정하는 문제를 정수계획법으로 정형화 한다. 먼저 첨자들은 다음과 같은 의미를 지닌다.

$i, j = 1,2,\ldots,N$ 작업을 나타내는 첨자

$k = 1,2,\ldots,K$ 로보트를 표시하는 첨자

또, t_{ik}는 i번째 작업을 k 번째 로보트에서 실행할 때의 소요 시간을 나타내며, 결정해야 할 변수들은 다음과 같다. 의사결정을 위한 정수 변수 (0 또는 1) X_{ik}, Y_{ij} 에 의해서 각 작업의 시작 시간 S_i 가 결정된다.

$X_{ik} = 1$ i번째 작업을 로보트 k가 처리하는 경우

$= 0$ 그 밖의 경우

$Y_{ij} = 1$ i번째 작업이 같은 로보트에서 j번째 작업보다 먼저 수행되는 경우

$= 0$ 그 밖의 경우

S_i i 번째 작업의 시작 시간

이들 변수들을 이용하여 정형화된 일정계획 문제는 다음과 같다.

$$\text{Minimize} \quad S_{N+1} + Q\left(\sum_i S_i\right) \tag{3.1}$$

Subject to

$$\sum_k X_{ik} = 1 \qquad i\text{가 non-RlaR2 작업일 때} \tag{3.2}$$

$$\sum_k X_{ik} = 2 \qquad i\text{가 RlaR2 작업일 때} \tag{3.3}$$

$$X_{ik} = 1 \qquad i\text{작업이 로보트 k에 할당된 경우} \tag{3.4}$$

$$S_i \geq 0 \qquad i = 1,2,\ldots,N \tag{3.5}$$

$$S_{N+1} - S_i \geq \sum_k X_{ik}t_{ik} \qquad i\text{가 non-RlaR2 작업일 때} \tag{3.6}$$

$$S_{N+1} - S_i \geq t_{ik} \qquad i\text{가 RlaR2 작업일 때} \tag{3.7}$$

$$S_j - S_i \geq \sum_k X_{ik}t_{ik} \tag{3.8}$$

non-RlaR2 작업 i가 j보다 먼저 실행되어야 할 때

$$S_j - S_i \geq t_{ik} \tag{3.9}$$

RlaR2 작업 i가 j보다 먼저 실행되어야 할 때

$$S_j - S_i + H(1-Y_{ij}) + H(1-X_{ik}) + H(1-X_{jk}) \geq t_{ik} \tag{3.10}$$

$$S_i - S_j + HY_{ij} + H(1-X_{ik}) + H(1-X_{jk}) \geq t_{jk} \tag{3.11}$$

$i \geq 1, \quad j \leq N, \quad 1 \leq k \leq K,$

i, j 간에 선후 관계가 지정되지 않은 경우

식 (3.1) 에 표시된 목적함수와 제약식 (3.6), (3.7) 은 모든 작업보다 뒤에 실행되어야 하는 가상의 N+1 번째 작업을 두어 이 작업의 시작 시간을 최소화함으로써 작업 종료 시간을 최소화하는 스케쥴을 얻고자 함을 보인다. 목적함수의 두번째항은 작업 종료시간에는 무관하지만 국소적인 left shift 를 없애기 위해 추가된 항으로, Q 값은 작업 수행에 소요되는 시간의 단위에 비해 무시할 수 있는 작은 값으로 한다. 식 (3.2) 는 non-RlaR2 작업이 하나의 로보트에 할당되어야 한다는 제약 조건을, 식 (3.3) 은 RlaR2 작업이 두개의 로보트에 할당되어야 함을 나타낸다. 작업의 특성상 특정 로보트에 할당된 작업은 식 (3.4) 로 나타나며, 식 (3.5) 는 각 작업의 시작 시간에 대한 하한값을 제공한다. 식 (3.8) 과 (3.9) 는 작업들간의 선후 관계에 의한 제약을 의미하는데, RloR2p 작업의 경우 어떤 로보트에 할당될지가 미정이므로 각 로보트에서의 소요시간에 의사결정변수 X_{ik} 를 곱하여 i 번째 작업의 소요시간을 구한다. 식 (3.10), (3.11) 은 두개의 작업이 같은 로보트에서 수행될 경우 이들간의 순서를 정하기 위한 제약식이다. 상수 H 는 t_{ik} 보다 매우 큰 값이므로 이들 제약식은 작업 i, j 가 같은 로보트에 할당된 경우만 의미를 가지며, 그렇지 않은 경우는 항상 성립하는 무의미한 제약식이 된다. 따라서 의사결정 변수 Y_{ij} 역시, 두 개의 작업이 같은 로보트에서 수행될 때만 위에서 정의한 의미로 해석될 수 있다.

작업간의 선후관계에 대한 제약이 전혀 없고, 특정 로보트에 할당된 작업이 없을 때, 위의 정수계획법에 이용되는 변수의 갯수는 실변수 N+1 개 (S_i), 0/1 변수 $N\cdot K+{}_NC_2$ 개 (X_{ik}, Y_{ij}) 이며, 제약식의 수는 $3\cdot N+2\cdot K\cdot {}_NC_2$ 개 이다. 그러나 조립 작업과 같은 대부분의 실제 응용에서는 작업들 사이의 선후 관계가 분명하게 주어지는데, 하나의 선후관계가 지정될 때마다 제약식 2·K 개, 변수 1 개가 감소하게 된다. 또한, 로보트의 기능이 다를 경우 특정 로보트만이 처리할 수 있는 작업이 존재하게 되는데, 특정 작업을 하나의 로보트에 고정시켜 할당할 때마다 제약식 2·N·(K-1)개, 변수 K개를 감소시키는 효과가 있다.

이와 같이 문제의 특징을 반영하여 제약식의 수 및 변수의 수를 줄여갈 수 있는 위의 정수계획법은 다음과 같은 특징을 갖는다.

- RlaR2 작업의 도입이 별도의 변수나 제약식을 요구하지 않는다.
- 로보트의 수가 하나 증가할 때, 변수의 갯수는 작업의 수만큼만 증가한다.
- 로보트들의 기능에 차이가 있는 경우도 동등한 경우에 비하여 복잡도가 증가하지 않는다.

3.3 시뮬레이션 결과

3.2 절에서 혼합 정수계획법으로 정형화한 문제를 간단한 조립작업에 적용하여 보았다. 시뮬레이션은 별도의 소프트웨어를 작성하지 않고, IBM PC용 정수계획법 팩키지 LINDO 를 이용하였다.

대상 작업의 조립 계획 (assembly plan) 을 그래프로 나타내면 그림 3.1 과 같다. 작업의 수는 8개이며, 빗금친 노드는 RlaR2 작업을, 흰 점이 표시된 노드는 로보트 1 에 할당된 작업을, 검은 점은 로보트 2에 할당된 작업을, 표시가 없는 노드는 RloR2p 작업임을 나타낸다. 로보트의 최대 갯수는 3대로 잡고, 표 3.1 에 각 로보트가 작업을 수행하는 데에 소요되는 시간을 표시하였다.

먼저 로보트 R1, R2만을 작업에 참여 시킬 경우, 각 작업의 시작 시간을 나타내는 9개의 변수외에 의사결정을 위한

0/1 변수 19 개가 이용되었으며, 제약식은 46 개가 필요하였다. 출력된 스케쥴을 그림 3.2 에 보인다. 같은 작업을 대상으로 로보트의 수를 3대로 스케쥴한 경우는 0/1 변수가 26 개, 제약식의 수가 57 개로 증가하였다. 결과는 그림 3.3 에 보인 바와 같다.

이와 같은 결과를 얻기까지 프로그램의 수행 시간은 IBM PC/AT 에서 3분정도 소요되어, 비교적 작은 규모의 문제는 3.2 절에서 정형화한 정수계획법으로 쉽게 최적의 스케쥴을 구할 수 있음을 보았다. 그러나 작업의 수가 증가함에 따라 식 (3.10), (3.11) 의 형태로 나타나지는 순서계획 (sequencing) 변수및 제약식이 급격히 증가하게 되므로, 문제의 규모가 큰 경우는 정수계획법 고유의 문제를 해결하기 위한 별도의 풀이 방안이 마련되어야 함을 알 수 있었다.

표 3.1 로보트별 작업 처리 시간

Table 3.1 Processing Time for Each Robot

	1	2	3	4	5	6	7	8
R1	2.5	3.4	2.0	1.1	3.0	4.0	-	1.5
R2	2.1	-	2.0	1.2	3.0	3.5	2.8	-
R3	2.6	-	2.0	1.5	2.7	4.2	-	-

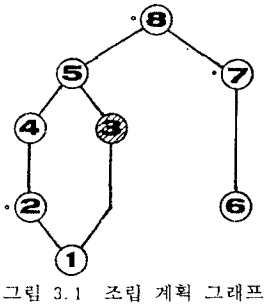

그림 3.1 조립 계획 그래프

Fig. 3.1 Assemble Plan

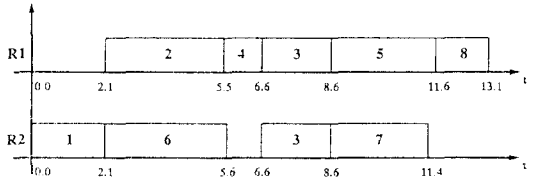

그림 3.2 8 작업 2 로보트의 예에 대한 일정계획

Fig. 3.2 Schedule for 8 Task 2 Robot Example

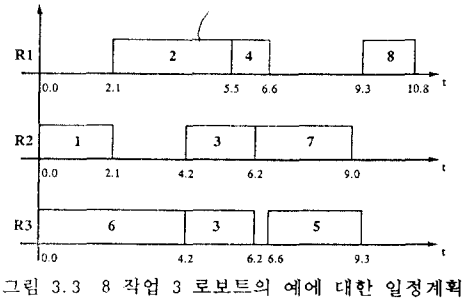

그림 3.3 8 작업 3 로보트의 예에 대한 일정계획

Fig. 3.3 Schedule for 8 Task 3 Robot Example

4. 간단한 오류 회복을 고려한 MRS 의 성능 평가

4.1 시스템 모델링

여기서 연구하고자 하는 시스템은 3가지 인데 그것들은 각각 시스템A, 시스템B, 시스템C 로 칭하며 그 시스템에 대한 모델을 모델A, 모델B, 모델C 라 한다. 그림 4.1은 시스템A,B,C의 구조를 나타낸다.

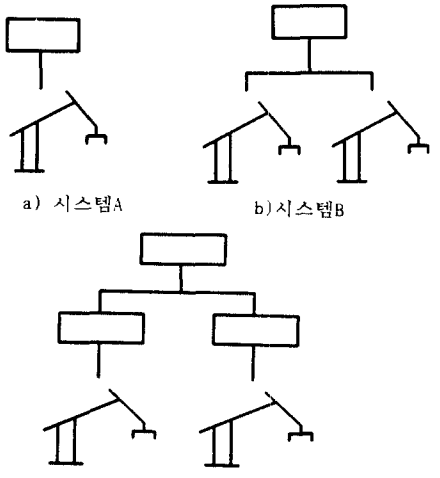

그림 4.1 시스템 A,B,C의 구조

Fig 4.1 Architecture of System A,B,C

4.1.1 시스템A 모델링

시스템A는 SRS(Single Robot System)이다. 이것은 MRS와 비교하기 위해 사용한다. 시스템 A의 큐잉 모델은 그림 4.2 에 나타나 있다.

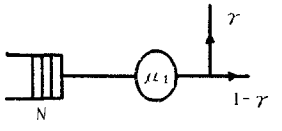

그림 4.2 시스템A의 큐잉 모델

Fig 4.2 Queueing Model of System A

4.1.2 시스템B 모델링

시스템B는 하나의 프로세서가 두대의 로보트를 구동하는 시스템이다. 프로세서 하나로 로보트 두대를 동작시키는 방법에는 크게 두가지로 나눌수 있다. 즉 프로세서 한개로 로보트대를 동시에 구동하는 것과 한번에 한대씩 구동하는 방법이 있을 수 있다. 물론 한번에 두대씩 동작 시키는 것이 좋으나 그렇게 되려면 로보트 두대분의 궤적 실행 (trajectory execution)을 실시간에 할 수 있을 만큼 프로세서의 성능이 충분해야 한다. 동시동작이 가능한 경우라도 각각의 로보트가 독립적인 행위자(server)로 작동하는 것이 아니라 두개가 동기화 되어 실행한다. 로보트의 갯수가 증가되면 동시동작 방법은 확장하는데 한계가 있다. 여기서는 전자의 방법 하나만 고려한다. 시스템 B의 큐잉 모델은 그림 4.3 에 나타나 있다.

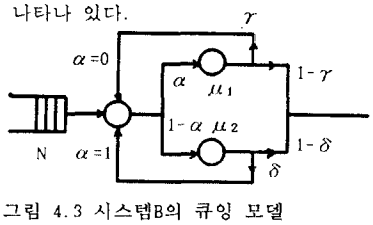

그림 4.3 시스템B의 큐잉 모델

Fig 4.3 Queueing Model of System B

4.1.3 시스템C 모델링

시스템C는 각각의 로보트가 자신의 프로세서를 가지고 있어서 로보트를 독립적으로 동작 시키고 각각 프로세서를 제어하는 또다른 프로세서가 있는 이른바 계층적 구조이다. 이러한 구조는 Alford and Belyeu[9], Luh and Zheng[10]등에 나타나 있는 바와 같이 MRS에서의 가장 많이 사용되는 구조이다. 프로세서간의 통신은 주프로세서가 통괄한다. 주프로세서에서 부프로세서로 명령을 이송할 경우에는 이송시간을 고려한다. 이송 시간은 일차함수로 표현할 수 있다. 즉 이송할 명령의 갯수에 비례하며 기본적으로 데이타 전송에 필요한 시간이 일괄적으로 첨가된다. 물론 로보트에서 오류가 생길 경우에는 부프로세서에서 주프로세서로 데이타 이송이 필요하다 이것도 마찬가지로 일차 함수로 표현한다. 단 주프로세서와 부프로세서가 서로 통신할 경우에는 해당 부프로세서는 명령 수행을 하지 못하나 다른 부프로세서는 독립적으로 명령을 수행할 수 있다. 식(4.1) 과 식(4.2)에 전송시간에 관한 식이 있으며 시스템 C의 큐잉 모델은 그림 4.4 에 나타나 있다.

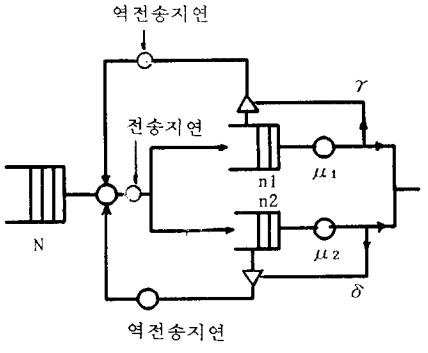

그림 4.4 시스템C의 큐잉 모델

Fig 4.4 Queueing Model of System C

$$f(n) = k1\ n + k2 \quad (k1,k2\text{는 전송 상수}) \tag{4.1}$$

$$g(n) = k3\ n + k4 \quad (k3,k4\text{는 역전송 상수}) \tag{4.2}$$

4.2. 수학적 분석

먼저 여기에서 공통으로 사용되는 표식 및 용어를 다음에 나타내었다.

- N : 주어진 명령어 갯수
- μ_1 : 로보트 1의 수행 시간 비율(service rate)
- μ_2 : 로보트 2의 수행 시간 비율(service rate)
- α : 명령 한개가 로보트1으로 전달되는 확률
- γ : 명령을 한번 수행할때 로보트1에 오류가 생길 확률
- δ : 명령을 한번 수행할때 로보트2에 오류가 생길 확률
- t_{com} : 전체 실행 완료 시간
- P_{sf} : 실행 실패가 일어날 확률
- P_{ab} : 로보트1,2 모두 오류 발생 없이 실행완료할 확률
- P_{all} : 로보트1,2가 동시에 오류가 발생할 확률
- $P_{\bar{a}b}$: 로보트1에서 오류가 발생했으나 로보트2에서 나머지를 실행완료할 확률
- $P_{\bar{b}a}$: 로보트2에서 오류가 발생했으나 로보트1에서 나머지를 실행완료할 확률
- $P_{\bar{a}\bar{b}}$: 로보트1에서 먼저 오류가 발생하여 나머지 명령을 로보트2로 옮겨 수행할때 다시 로보트2에서 오류가 발생하는 확률
- $P_{\bar{b}\bar{a}}$: 로보트2에서 먼저 오류가 발생하여 나머지 명령을 로보트1로 옮겨 수행할때 다시 로보트1에서 오류가 발생하는 확률
- C_{rm} : 실행 실패가 발생하여 실행 완료하지 못한 명령 갯수의 기대값
- C_{ab} : 확률 $P_{\bar{a}b}$가 생기는 경우에 실행 완료하지 못한 명령 갯수의 기대값
- C_{ba} : 확률 $P_{\bar{b}a}$가 생기는 경우에 실행 완료하지 못한 명령 갯수의 기대값

실행 실패(service failure)란 주어진 명령을 전부 실행 완료하지 못하는 경우를 말하며 그확률을 실행 실패 확률이라 한다. 시스템A 경우에는 로보트가 한대 밖에 없으므로 오류(error)가 생기면 그대로 실행 실패가 되나 시스템B나 시스템C는 하나의 로보트에 오류가 생기더라도 다른 로보트에서 실행완료 시킬 수 있다.

평균 실행 완료 시간(average completion time)은 실행 실패가 없는 경우 주어진 명령을 전부 수행하는데 걸리는 시간의 기대치이다.

4.2.1 모델A의 분석

모델A의 t_{com}, P_{sf}, C_{rm}은 다음 식(4.3) 에서 식(4.5)에 나타나 있다.

$$t_{com} = N \frac{1}{\mu} \tag{4.3}$$

$$P_{sf} = \sum_{i=0}^{N-1} \gamma(1-\gamma) = 1 - (1-\gamma)^N \tag{4.4}$$

$$C_{rm} = \sum_{i=0}^{N-1} (1-\gamma)^i \gamma(N-i) \tag{4.5}$$

4.2.2 모델B의 분석

모델B에서 사용되는 표식에는 다음과 같은 것들이 있다.

- $P\mu_1$: 명령들이 로보트1을 통해 실행 완료할 확률
- $P\mu_2$: 명령들이 로보트2를 통해 실행 완료할 확률
- ρ : 실행 실패가 없을 경우 로보트1을 통해 실행 완료할 확률
- q : 실행 실패가 없을 경우 로보트2을 통해 실행 완료할 확률

모델B에서 로보트1,2 모두 오류 없이 명령 i개를 실행 완료할 확률을 P(i)로 나타내면

$$P(i) = \{\alpha(1-\gamma)+(1-\alpha)(1-\delta)\}^i \tag{4.6}$$

그러므로 $P_{ab} = P(N)$ 이 된다.

$$P_{ab} = \{\alpha(1-\gamma)+(1-\alpha)(1-\delta)\}^N \tag{4.7}$$

t_{com}을 구하기 위해 $P_{\bar{a}b}$, $P_{\bar{b}a}$, $P\mu_1$, $P\mu_2$, ρ, 등을 구하면 아래의 식과 같다.

$$P_{\bar{a}b} = \alpha\gamma \sum_{i=0}^{N-1} (1-\delta)^{N-i} P(i) \tag{4.8}$$

$$P_{\bar{b}a} = (1-\alpha)\delta \sum_{i=0}^{N-1} (1-\gamma)^{N-i} P(i) \tag{4.9}$$

$$P\mu_1 = \alpha(1-\gamma)P(N-1) + P_{\bar{b}a} \tag{4.10}$$

$$P\mu_2 = (1-\alpha)(1-\delta)P(N-1) + P_{\bar{a}b} \tag{4.11}$$

$$\rho = \frac{P\mu_1}{P\mu_1 + P\mu_2} \tag{4.12}$$

$$q = \frac{P\mu_2}{P\mu_1 + P\mu_2} \tag{4.13}$$

이 모델은 2-종류 평행 서버 (2-stage parallel server) 이므로 명령 한개의 평균 수행 시간은 다음 식(4.14)과 같다.

$$t_{avg} = \frac{q\cdot\mu_1 + \rho\cdot\mu_2}{\mu_1\cdot\mu_2} \tag{4.14}$$

$$t_{com} = N t_{avg} + \frac{P_{\bar{b}a}}{(P\mu_1+P\mu_2)\mu_2} + \frac{P_{\bar{a}b}}{(P\mu_1+P\mu_2)\mu_1} \tag{4.15}$$

P_{sf}와 C_{rm}을 구하기 위해서 $P_{\bar{a}\bar{b}}$, $P_{\bar{b}\bar{a}}$, C_{ab}, C_{ba}등을 구한다.

$$P_{\bar{a}\bar{b}} = \alpha\gamma\delta \sum_{i=0}^{N-1} [P(i) \sum_{j=0}^{N-i-1} [(1-\delta)^j]] \tag{4.16}$$

$$P_{\bar{a}\bar{b}} = (1-\alpha)\delta\gamma \sum_{i=0}^{N-1}[P(i) \sum_{j=0}^{N-i-1}[(1-\gamma)^j]] \quad (4.17)$$

$$P_{af} = P_{a\bar{b}} + P_{\bar{a}\bar{b}} \quad (4.18)$$

$$C_{ab} = \alpha\gamma\delta \sum_{i=0}^{N-1}[P(i) \sum_{j=0}^{N-i-1}[(1-\delta)^j(N-i-j)]] \quad (4.19)$$

$$C_{ba} = (1-\alpha)\delta\gamma \sum_{i=0}^{N-1}[P(i) \sum_{j=0}^{N-i-1}[(1-\gamma)^j(N-i-j)]] \quad (4.20)$$

$$C_{rm} = C_{ab} + C_{ba} \quad (4.21)$$

4.2.3 모델 C의 분석

모델C에서 사용하는 표식을 정리하면 아래와 같다.

n1 : 주프로세서에서 로보트1의 프로세서로 이송된 명령의 갯수

n2 : 주프로세서에서 로보트2의 프로세서로 이송된 명령의 갯수

그러므로 n1과 n2의 합은 N이 된다.

P_{all} : 로보트1,2가 동시에 오류가 발생하는 확률

$n_{bs}(i)$: 로보트1에서 i개의 명령이 오류 없이 실행될 때 로보트2에서 실행될 명령 갯수의 기대값

$$n_{bs}(i) = \min\{ \mathrm{INT}(i \frac{\mu_2}{\mu_1}), n2 \} \quad (4.22)$$

$n_{as}(i)$: 로보트2에서 i개의 명령이 오류 없이 실행될 때 로보트2에서 실행될 명령 갯수의 기대값

$$n_{as}(i) = \min\{ \mathrm{INT}(i \frac{\mu_1}{\mu_2}), n1 \} \quad (4.23)$$

n_{atb} : 로보트1에서 먼저 오류가 생길 경우 주프로세서로 전송해야 할 명령 갯수의 기대값

$$n_{atb} = \gamma \sum_{i=0}^{n1-1}(n1-i)(1-\gamma)^i(1-\delta)^{n_{bs}(i+1)} \quad (4.24)$$

n_{bta} : 로보트1에서 먼저 오류가 생길 경우 주프로세서로 전송해야 할 명령 갯수의 기대값

$$n_{bta} = \delta \sum_{i=0}^{n2-1}(n2-i)(1-\delta)^i(1-\gamma)^{n_{as}(i+1)} \quad (4.25)$$

t_{ae} : 로보트1에서 먼저 오류가 생길때까지 걸린 시간의 기대값

$$t_{ae} = \frac{n1 - n_{atb}}{\mu_1} \quad (4.26)$$

t_{be} : 로보트2에서 먼저 오류가 생길때까지 걸린 시간의 기대값

$$t_{be} = \frac{- n_{bta}}{\mu_2} \quad (4.27)$$

$t_{com/ab}$: 확률 P_{ab}일 경우 전체 실행 기산의 기대치

$t_{com/\bar{a}b}$: 확률 $P_{\bar{a}b}$일 경우 전체 실행 기산의 기대치

$t_{com/b\bar{a}}$: 확률 $P_{b\bar{a}}$일 경우 전체 실행 기산의 기대치

위의 표식을 이용하여 $t_{com/ab}$, $t_{com/\bar{a}b}$, $t_{com/b\bar{a}}$를 구하면 다음과 같다.

$$P_{ab} = (1-\gamma)^{n1} \quad (1-\delta)^{n2} \quad (4.28)$$

$$t_{com/ab} = k1\cdot N + k2 + \max\{ n1/\mu_1 , n2/\mu_2 \} \quad (4.29)$$

$$P_{\bar{a}b} = \gamma(1-\delta)^{n2} \sum_{i=0}^{n1-1}(1-\gamma)^i(1-\delta)^{n1-i} \quad (4.30)$$

만일 로보트1에서 오류가 먼저 발생했을때 로보트2가 휴지상태(idle state) 이면 실행 완료 시간에 역전송 시간이 포함되어야 한다. 그러므로 $t_{ae} + k3\cdot n_{atb} + k4$가 $n2/\mu_2$ 보다 크면

$$t_{com/\bar{a}b} = k1\cdot N+k2+t_{ae}+k3\ n_{atb}+k4+k1\cdot n_{atb}+k2+n_{atb}/\mu_2 \quad (4.31)$$

그렇지 않을 경우에는

$$t_{com/\bar{a}b} = k1\cdot N+k2+\frac{n2 + n_{atb}}{\mu_2} + k1\cdot n_{atb}+k2 \quad (4.32)$$

$$P_{b\bar{a}} = \delta(1-\gamma)^{n1} \sum_{i=0}^{n2-1}(1-\delta)^i(1-\gamma)^{n2-i} \quad (4.33)$$

또한 만일 로보트2에서 오류가 먼저 발생했을때 로보트2가 휴지상태(idle state) 이면 실행 완료 시간에 역전송 시간이 포함되어야 한다. 그러므로 $t_{be} + k3\cdot n_{bta} + k4$가 $n1/\mu_1$ 보다 크면

$$t_{com/b\bar{a}} = k1\cdot N+k2+t_{be}+k3\ n_{bta}+k4+k1\cdot n_{bta}+k2+n_{bta}/\mu_1 \quad (4.34)$$

그렇지 않을 경우에는

$$t_{com/b\bar{a}} = k1\cdot N+k2+\frac{n1 + n_{bta}}{\mu_1} + k1\cdot n_{bta}+k2 \quad (4.35)$$

$$t_{com} = \frac{P_{ab}\ t_{com/ab}}{P_{ab}+P_{\bar{a}b}+P_{b\bar{a}}} + \frac{P_{\bar{a}b}\ t_{com/\bar{a}b}}{P_{ab}+P_{\bar{a}b}+P_{b\bar{a}}} + \frac{P_{b\bar{a}}\ t_{com/b\bar{a}}}{P_{ab}+P_{\bar{a}b}+P_{b\bar{a}}} \quad (4.36)$$

Psf와 Crm을 구하는 과정이 아래 식(4.37)에서 식(4.44)에 나타나 있다.

$$P_{all} = \gamma\delta \sum_{i=0}^{n1-1}(1-\gamma)^i(1-\delta)^{n_{bs}(i)} \quad (4.37)$$

$$P_{ab} = \gamma\delta \sum_{i=0}^{n1-1}[(1-\gamma)^i(1-\delta)^{n_{bs}(i+1)} \sum_{j=0}^{N-i-n_{bs}(i+1)-1}[(1-\delta)^j]] \quad (4.38)$$

$$P_{ba} = \delta\gamma \sum_{i=0}^{n2-1}[(1-\delta)^i(1-\gamma)^{n_{as}(i+1)} \sum_{j=0}^{N-i-n_{as}(i+1)-1}[(1-\gamma)^j]] \quad (4.39)$$

$$P_{af} = P_{all} + P_{\bar{a}\bar{b}} + P_{b\bar{a}} \quad (4.40)$$

$$C_{all} = \gamma\delta \sum_{i=0}^{n1-1}[(1-\gamma)^i(1-\delta)^{n_{bs}(i)}\ (N-n_{bs}(i)-i)] \quad (4.41)$$

$$C_{ab} = \gamma\delta \sum_{i=0}^{n1-1}[(1-\gamma)^i(1-\delta)^{n_{bs}(i+1)} \sum_{j=0}^{N-i-n_{bs}(i+1)-1}[(1-\delta)^j(N-n_{bs}(i)-i-j)]] \quad (4.42)$$

$$C_{ba} = \delta\gamma \sum_{i=0}^{n2-1}[(1-\delta)^i(1-\gamma)^{n_{as}(i+1)} \sum_{j=0}^{N-i-n_{as}(i+1)-1}[(1-\gamma)^j(N-n_{as}(i)-i-j)]] \quad (4.43)$$

$$C_{rm} = C_{all} + C_{ab} + C_{ba} \quad (4.44)$$

4.2.4 적용 실례

실제로 구체적인 값에 대하여 계산해 보고 그결과를 분석해 본다. 여기서 사용한 값은 γ=0.001, δ=0.001, N = 1000, $\mu_1=\mu_2$=1, α=0.5 이며 모델C에서 n1과 n2의 비는 1:1로 하였다. 그림 4.5 에서 보면 모델 A,B는 실행 완료 시간이 거의 같고 모델 C는 모델 A의 반 정도이다.그림 4.6,그림 4.7 은 MRS의 오류 회복 성질을 나타내며 그림 4.8 과 그림 4.9 는 모델 C 에 전송 시간이 있을 경우이다. 그림 4.8 은 전송 시간이 전송 데이타량과 관계없는 경우이고 그림 4.9 는 영향을 주는 경우이다.

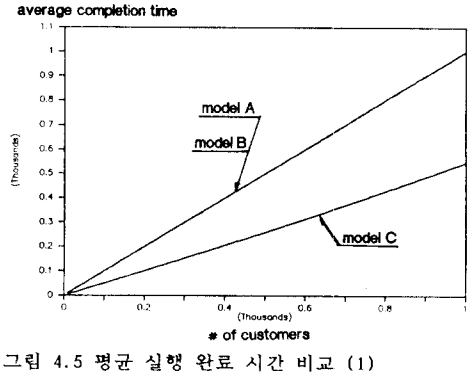

그림 4.5 평균 실행 완료 시간 비교 (1)

Fig 4.5 Comparison of Average Completion Time (1)

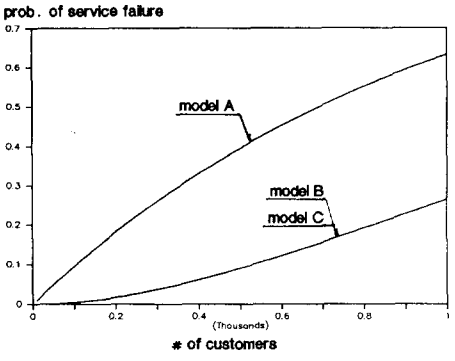

그림 4.6 실행 실패 확률 비교

Fig 4.6 Comparison of Service Failure

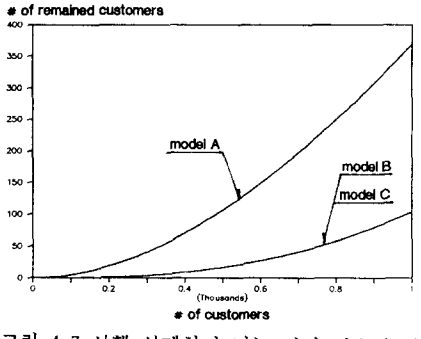

그림 4.7 실행 실패하여 남는 명령 갯수의 기대값

Fig 4.7 Expectation Value of Remained Customers

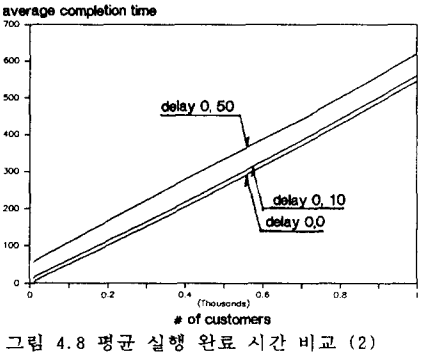

그림 4.8 평균 실행 완료 시간 비교 (2)
(modelC, k1=k3=0, k2=k4=0, 10, 50)

Fig 4.8 Comparison of Average Completion Time (2)

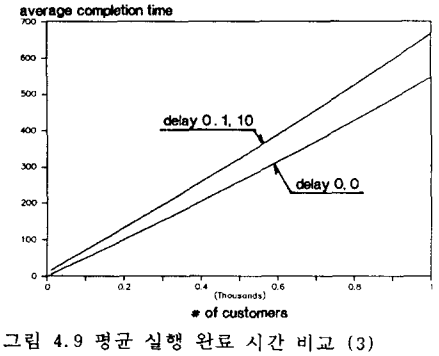

그림 4.9 평균 실행 완료 시간 비교 (3)
(modelC, k1=k3=0, k2=k4=10 and k1=k3=0.1, k2=k4=10)

Fig 4.9 Comparison of Average Completion Time (3)

5. 결론

본 논문에서는 MRS 작업 스케쥴링과 수식적 성능 평가의 두부분으로 대별할 수 있다. 작업 스케쥴링에서는 로보트의 다양성과 유연성을 고려하여 부품의 경로와 순서 계획을 동시에 처리하는 일정계획 문제를 정수계획법으로 정형화하였다. 간단한 작업을 대상으로는 최적해를 구할 수 있었지만, 좀 더 많은 작업을 대상으로 하는 실제적인 일정계획을 위해서는 정수계획문제를 다루는 효과적인 알고리즘의 개발이 필요하며, 도구 교환에 의한 영향 및 다양한 목적 함수를 도입하기 위한 연구가 수행되어져야 한다.

수식적 성능 평가에서는 오류가 생길 경우 회복하는 기능을 가진, 간단한 독립 작업을 하는 두 로보트 경우에 대해서 성능 평가를 하였다. 여기서 살펴본 MRS 구조는 두가지 대표적인 것에 대해 분석하였으나 다른 구조에 대해서도 분석할 필요가 있으며 로보트도 두대가 아니고 여러대로 확장하여 해석할 수 있으며 작업의 종류도 앞에서 제시한 여러가지 경우에 대해서 조사하고 나아가서 컴퓨터 시뮬레이션이 필요하다. 본연구는 이러한 실제적인 MRS 를 분석하기 위한 기본적인 도구가 될 수 있을 것이다. 나아가서 작업 스케쥴링을 고려한 MRS 성능 평가를 할 수 있을 것이다.

참고문헌

[1] Kennet R. Baker, Introduction to Sequencing and Scheduling, John Wiley & Sons, Inc., 1974.

[2] Eng-Joo Lee and Pitu B. Mirchandani, "Concurrent Routing, Sequencing, and Setups for a Two-Machine Flexible Manufacturing Cell," IEEE J.of Robotics and Automation, Vol.4, No.3, 1988.

[3] B.L.Bodnar and A.C.Liu Modeling and Performance Analysis of Single-Bus Tightly-Coupled Multiprocessors", IEEE Trans. Comput., Vol.38, No.3, Mar.1989

[4] K.O.Siomalas and B.A.Bowen Performance of Cross-Bar Multiprocessor Systems", IEEE Trans. Comput., Vol.C-32, No.7, Jul.1983

[5] K.B.Irani and I.H.Onyuksel "A Closed-Form Solution for the Performance Analysis of Multiple-Bus Multiprocessor systems", IEEE Trans. Comput., Vol.C-33, No.11, Nov.1984

[6] J.Y.Han and C.Y.Wang Modeling and Performance Evaluation of Multiprocessor Systems for Real-Time Nonlinear Robot Control", IEEE Conf. R&A 1989

[7] D.Coupez, A.Delchambre and P.Gaspart, "The Supervision and Management of a Two Robots Flexible Assembly Cell," IEEE Int.Conf. on Robotics and Automation, 1989.

[8] Wilbert E. Wilhelm, "Complexity of Sequencing Tasks in Assembly Cells Attended by One or Two Robots," Naval Research Logistics, Vol.34, 1987.

[9] C.O.Alford and S.M.Belyeu "Coordinated Control of Two Robot Arms", IEEE Conf. R&A 1984

[10] J.Y.S.Luh and Y.F.Zheng "An Interactively Hierarchical Control Scheme for Two Coordinating Industrial Robots", Proceedings of 25th CDC Dec.1986