한국시뮬레이션학회 논문지
Vol. 20, No. 4, pp. 59-66 (2011. 12)

시뮬레이티드 어닐링 기반 m-RUN 교착 회피 정책 생성 알고리즘 설계

최진영[1†]

Design of an Algorithm for Generating m-RUN Deadlock Avoidance Policy Based on Simulated Annealing

Jin Young Choi

ABSTRACT

This work presents an algorithm for generating multi-RUN (m-RUN) deadlock avoidance policy based on simulated annealing algorithm. The basic idea of this method is to gradually improve the current m-RUN DAP after constructing an initial m-DAP by using simple m RUN DAPs. The search for a neighbor of the current m-RUN DAP is done by selecting and changing only one component of the current m-RUN, while accepting some unimproved solutions with some probability. It is examined for its performance by generating some sample system configurations.

Key words : Simulated annealing, Multi resource upstream neighborhood (m-RUN) deadlock avoidance policy

요 약

본 연구에서는 시뮬레이티드 어닐링 알고리즘에 기반한 다중 RUN(multi-RUN: m-RUN) 교착 제어 정책 생성 알고리즘 설계에 대해 제안하였다. 이 방법은 단순한 RUN DAP를 m개 생성한 후 이들의 합성에 의해 초기 m-RUN DAP를 정의하고 이를 점차적으로 개선시켜 나가는 것이다. 이 때 이웃(Neighbor) m-RUN은 현재 m-RUN에서 오직 한 개의 성분 RUN만을 랜덤하게 수정하여 생성하는 지역 탐색 기법을 적용하여 선택하였다. 또한 몇 가지 기본적인 시스템 구성을 가정하고 수치 실험을 적용하여 제안된 교착 제어 정책 성능의 우수성을 평가하였다.

주요어 : 시뮬레이티드 어닐링, 다중 RUN 교착 회피 정책

1. 서 론

최근 들어 첨단 제조시스템에서는 생산성을 증대시키면서 동시에 유연성을 확보하여 각각의 공정간 재공품(Work-In-Process: WIP)의 원활한 흐름을 제공하면서 전체 시스템의 효율을 극대화 시켜야 하는 필요성이 점차 증가하고 있다. 특히, 반도체 및 LCD 산업에서는 나노팹(Nano Fab) 기술 등의 도입과 함께 모든 생산 공정이 점차로 초소형화 및 완전 자동화 되어 감에 따라 각 공정의 처리를 위해서 필요한 고가의 워크스테이션 등의 자원을 효율적으로 스케쥴링하고 관리하는 문제가 전체 시스템의 성능에 중요한 영향을 미치고 있다.

예를 들면 그림 1과 같이 세 개의 워크스테이션 R_1, R_2, R_3와 한 개의 트랜스포터로 구성되어 있는 자동화 제조 시스템에서 두 가지 타입의 제품 J_1, J_2가 각각 서로

접수일(2011년 9월 20일), 심사일(1차 : 2011년 11월 30일), 게재 확정일(2011년 11월 30일)
[1)] 아주대학교 산업정보시스템공학부
주 저 자 : 최진영
교신저자 : 최진영
E-mail; choijy@ajou.ac.kr

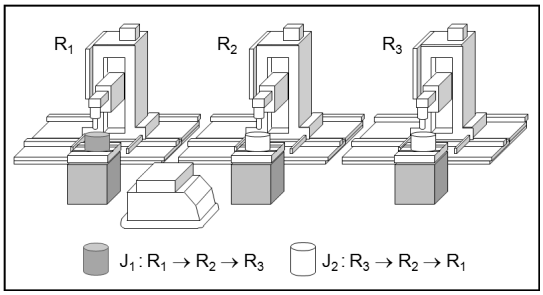

그림 1. 자동화 제조 시스템 교착 문제의 예

다른 방향의 처리 공정으로 만들어진다면, 워크스테이션과 트랜스포터의 버퍼 용량 유한성으로 인해 처리되는 job들 간에 심각한 블록킹 현상이 발생할 수도 있게 된다. 만일, 버퍼 용량이 1일 때 두 가지 제품이 동시에 시스템에 로딩되어 그림 1과 같은 상태가 된다면 J_1과 J_2는 더 이상 처리될 수 없는 상태가 되고 시스템은 완전히 정지된다(Choi 등, 2003).

자원 할당 시스템에서의 교착문제(Deadlock problem)는 이러한 현상을 정의하며, 지금까지 이를 해결하기 위한 노력으로 논리 제어 정책(Logical Control Policy)에 관한 연구가 다음과 같이 세 가지 분야로 나누어져 많이 진행되어 오고 있다(Reveliotis 등 1997; Reveliotis, 2005).

- 교착 방지(Deadlock Prevention): 교착이 발생하지 않도록 시스템 설계 단계에서 시스템 자원에 많은 제한을 가하는 방법
- 교착 탐지 및 복구(Deadlock Detection and Recovery): 시스템을 운영하는 중에 교착 문제가 발생하면 탐지하여 복구하는 방법
- 교착 회피(Deadlock Avoidance): 시스템의 현재 상태와 미래 가능한 상태들에 대한 정보를 유지하면서 교착 문제가 발생할 가능성을 회피해 가는 방법

이 중에서 교착 회피 문제는 시스템을 운영하는 과정에서 발생할 수 있는 교착 문제를 실시간으로 제어할 수 있는 방안을 제공하는 것을 목표로 하며, 3가지 분야 중에서 가장 많은 연구 및 응용이 이루어지고 있다. 이 때, 시스템 동작의 유연성을 최대한 확보하기 위해서는 최적의(Optimal) 논리 제어 알고리즘을 적용하는 것이 중요한데, 이를 위해서는 시스템에서 존재할 수 있는 모든 상태(State)를 고려하여 논리적으로 안전한(Safe) 상태를 찾는 것이 필요하다. 여기서 논리적으로 안전한 시스템 상태란 시스템 내에 존재하는 모든 job들을 끝까지 완성하는데 필요한 모든 자원이 가용한 상태를 말한다. 그러나 일반적으로 시스템 상태의 수는 시스템 구성이 복잡해짐에 따라 기하급수적으로 증가하며, 이러한 이유로 최적의 교착 회피 정책(Deadlock Avoidance Policy: DAP)를 찾는 것은 NP-Complete한 문제로 알려져 있다(Reveliotis 등, 2001). 따라서 대부분의 교착 회피 정책에 관한 연구는 polynomial 시간 안에 가능하면 많은 수의 시스템 상태에 대한 안전성을 확인할 수 있는 알고리즘을 찾는 주제로 제한되어 수행되어져 오고 있다.

그 중에서 제품 생산 프로세스의 흐름을 고려하여 하향 워크스테이션이 상향 워크스테이션에 대한 완충 작용을 할 수 있도록 시스템 운용을 제어하는 교착 제어 정책을 Resource Upstream Neighborhood(RUN) 정책이라고 한다(Reveliotis, 2005). RUN DAP는 Gaarder(1993)에 의해 처음 아이디어가 제안되었으나, Reveliotis 등(1996)이 교착 정책 논리를 보다 명확하게 정의하고 알고리즘의 정확성을 수학적으로 증명하였으며, 버퍼 용량뿐 아니라 자원 순서(Resource ordering)에 기반 한 정책으로 일반화하였다. 이 방법은 시스템 상태의 안전성을 제품 처리 경로에 대한 시스템 자원의 완충 작용 관계를 나타내는 행렬과 버퍼 용량 또는 자원 순서를 나타내는 벡터를 이용하여 표현된 부등식으로 간단하게 확인할 수 있는 장점이 있으나, 지금까지 알려진 다른 DAP들에 비해 찾을 수 있는 안전 상태의 수에 의해 정의되는 알고리즘의 성능이 상대적으로 높지 않다는 것이 단점이다(Reveliotis, 2005).

본 연구에서는 이러한 RUN DAP의 성능을 향상시킬 수 있는 방법으로 시뮬레이티드 어닐링 알고리즘(Hillier 등, 2010)에 기반한 다중 RUN(multi-RUN: m-RUN) 교착 제어 정책 설계에 대해 제안하였다. 이 방법은 단순한 RUN DAP를 m개 생성한 후 이들의 합성에 의해 초기 m-RUN DAP를 정의하고 이를 점차적으로 개선시켜 나가는 것이다. 본 연구에서는 이를 위해서 시뮬레이티드 어닐링 방법을 이용하였으며, 이 때 이웃(Neighbor) m-RUN은 현재 m-RUN에서 오직 한 개의 성분 RUN만을 랜덤하게 수정하여 생성하는 지역 탐색 기법을 적용하여 선택하였다. 또한 몇 가지 기본적인 시스템 구성을 가정하고 수치 실험을 적용하여 제안된 교착 제어 정책 성능의 우수성을 평가하였고, m 값에 따른 m-RUN 정책의 성능 변화도 관찰하여 적절한 m 값을 찾기 위한 분석도 실행하였다. 그 결과 버퍼 용량을 사용하는 단순 RUN DAP의 성능을 크게 개선시킬 수 있었으며, m-RUN DAP의 성능에 크게 영향을 주는 최소한의 순서 벡터들이 존재하며 이들을 효율적 활용이 m-RUN DAP의 생성에 영향을 줄 수 있음을 실험적으로 확인하였다.

본 논문의 구성은 다음과 같다. 먼저 2장은 m-RUN DAP의 개념 및 적용 방법을 설명한다. 3장은 시뮬레이티드 어닐링 기반 m-RUN DAP 생성 알고리즘 설계에 대해 설명한다. 4장은 성능 분석을 위한 실험 설계 및 분석에 대해 다루며 마지막으로 5장에서 결론을 제시한다.

2. m-RUN 교착 회피 정책

2.1 RUN 교착 회피 정책

주어진 시스템에서 하나의 워크스테이션 R에 대한

upstream neighborhood는 그 워크스테이션 R에서 실행되는 모든 job stages와 다음과 같은 특정한 조건을 만족하는 선행 job stages로 구성된다.: 워크스테이션 R에서 처리되는 각각의 job stage에 대해 첫 번째 선행 job stage부터 시작하여 각 선행 job stage를 처리하는 워크스테이션 버퍼 용량이 워크스테이션 R의 버퍼 용량보다 작거나 같은 경우에만 선택되며, 워크스테이션 R보다 큰 버퍼 용량을 갖는 첫 번째 워크스테이션에서의 선행 job stage부터 그 이후의 모든 선행 job stages들은 제외된다.

예를 들면 그림 1에서 J_1은 $R_1 \rightarrow R_2 \rightarrow R_3$의 순서로 처리되는데 워크스테이션 R_2에 대한 upstream neighborhood는 R_2에서 처리되는 J_{12}와 이의 선행 job stage인 J_{11}이 된다(여기서 J_{1k}는 제품 J_1의 k번째 처리 단계를 의미함). 그러나 만일 R_1의 버퍼 용량이 R_2의 버퍼 용량보다 크다면 J_{11}은 선택되지 않는다(만일 J_{11} 이후에 선행 job stages가 있다면 그들도 선택되지 않음).

이러한 upstream neighborhood의 정의를 바탕으로 RUN DAP는 각각의 워크스테이션에 대해 그 upstream neighborhood에 있는 job들의 합이 그 워크스테이션의 버퍼 용량을 초과하지 않는 시스템 상태를 안전 상태로 허용하는 방법이다. 따라서 이 조건을 만족하는 시스템 상태는 각각의 워크스테이션에 대해 모든 upstream neighborhood에 있는 job들이 해당 워크스테이션으로 이동해 와도 안전한 상태를 보장할 수 있게 된다.

RUN 교착 회피 정책은 upstream neighborhood를 나타내는 행렬을 이용하여 다음과 같이 표현될 수 있다.: r개의 워크스테이션 $R_i (i = 1, \cdots, r)$ 로 구성된 시스템에서 p가지 종류의 제품이 각각 n_j개의 작업 단계로 정해진 처리 공정에 의해 생산된다고 할 때, 제품 j의 k번째 작업은 $J_{jk} (j = 1, \cdots, p, k = 1, \cdots, n_j)$로 표현되며 다음 부등식을 만족하는 시스템 상태 s는 안전하다.

$$A \bullet s \leq b$$

여기서 A는 upstream neighborhood를 나타내는 $r \times \sum_{j=1}^{p} n_j$ 행렬, s는 시스템 상태를 나타내는 벡터, b는 워크스테이션의 버퍼 용량을 나타내는 벡터이다. 그림 1의 경우 행렬 A는 다음과 같이 표현될 수 있다.

$$A = \begin{array}{c} \\ R1 \\ R2 \\ R3 \end{array} \begin{array}{c} \begin{array}{cccccc} J_{11} & J_{12} & J_{13} & J_{21} & J_{22} & J_{23} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

여기서 $A_{R_i, J_{ab}} = 1$은 워크스테이션 R_i에서 처리되는 작업 J_{ab}가 upstream neighborhood로 선택되었음을 의미한다. 예를 들면, 앞의 예에서 워크스테이션 R_2에 대한 upstream neighborhood인 J_{12}와 J_{11}에 대해 행렬 A는 $A_{R_2, J_{12}} = A_{R_2, J_{11}} = 1$의 값으로 표현된다.

이러한 RUN DAP의 결과는 벡터 b를 r개의 워크스테이션 간의 순서(ordering)를 나타내는 벡터로 대체하였을 경우에도 동일하게 성립함이 증명되었다(Reveliotis, 2005). 이 때 하나의 워크스테이션 R에 대한 upstream neighborhood를 판정할 때 버퍼 용량 대신 워크스테이션에 대응되는 순서 값을 이용한다.

2.2 m-RUN 교착 회피 정책

본 연구에서 제안한 m-RUN 교착 회피 정책은 단일 RUN DAP를 m개 생성한 후 이들의 합성에 의해 다음과 같이 정의된다.

Definition: (m-RUN 교착 회피 정책 π)

A_i와 b_i를 i번째 생성된 RUN DAP의 upstream neighborhood 행렬과 순서 벡터라고 할 때 m-RUN DAP 는

$$\pi = \bigvee_{i=1}^{m} (A_i \bullet s \leq b_i)$$

로 정의되며, m개의 선형부등식 중에 적어도 하나를 만족시키는 시스템 상태 s는 교착 현상이 발생하지 않는 안전한 상태이다.

위 정의에서 $\vee$ 기호는 고려되는 여러 조건 중에 적어도 하나의 조건이 만족되면 참이 되는 논리합을 의미한다. 이렇게 정의된 m-RUN DAP π는 다음과 같은 특성을 갖는다.

Property 1: 두 개 이상의 RUN DAP π를 합성해서 만든 m-RUN DAP는 올바른 DAP이며, 구성요소인 단일 RUN DAP들보다 적어도 더 적지 않은 안전 상태를 찾을 수 있다.

(증명) $m = 2$인 경우를 고려하면 상태 s가 $A_1 \bullet s \leq b_1$을 만족하고, $A_2 \bullet s \leq b_2$는 만족하지 않더라도 상태 s는 안전 상태이다. 같은 이유로 $m = q$일 때의 m-RUN DAP가 올바른 DAP라고 가정하면, $m = q+1$일 때의 m-RUN DAP도 올바른 DAP이다. 따라서 m-RUN DAP

가 찾을 수 있는 안전 상태 집합은 m-RUN DAP를 구성하는 각각의 RUN DAP가 찾을 수 있는 안전 상태 집합의 합집합이 되며, 그 크기는 각 구성요소에 의한 안전 상태 집합 보다 크거나 같다.

Property 2: 일반적으로 r개의 워크스테이션으로 구성된 시스템에 대해서 워크스테이션에 대응되는 순서 값을 적용하면 $r!$개의 RUN DAP와 ${}_{r!}C_m$개의 m-RUN DAP 생성이 가능하다.

(증명) r개의 워크스테이션으로 만들 수 있는 순서(ordering)를 나타내는 벡터 b는 $r!$가지가 있으며 이 중에서 m개를 선택하는 방법은 ${}_{r!}C_m$가지이다.

2.3 m-RUN DAP의 적용 예

그림 1의 시스템에 대해 워크스테이션에 대응되는 순서 값을 적용하여 m-RUN DAP를 적용하면 대상 시스템은 3개의 워크스테이션으로 구성되어 있으므로 모두 $3!=6$ 가지의 순서 벡터 b의 생성이 가능하다. 만일 $m=2$이고, 선택된 2개의 순서 벡터가 $b_1=\begin{bmatrix}3\\2\\1\end{bmatrix}, b_2=\begin{bmatrix}2\\3\\1\end{bmatrix}$이라고 가정하면 2-RUN DAP는 다음과 같이 정의된다.:

$$\begin{array}{c} \\ R_1 \\ R_2 \\ R_3 \end{array}\begin{array}{c} \begin{matrix} J_{11} & J_{12} & J_{13} & J_{21} & J_{22} & J_{23} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array} \bullet\ s \le \begin{bmatrix}3\\2\\1\end{bmatrix} \text{ 또는}$$

$$\begin{array}{c} \\ R_1 \\ R_2 \\ R_3 \end{array}\begin{array}{c} \begin{matrix} J_{11} & J_{12} & J_{13} & J_{21} & J_{22} & J_{23} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array} \bullet\ s \le \begin{bmatrix}2\\3\\1\end{bmatrix}$$

첫 번째 부등식에서 세 워크스테이션에 대응되는 벡터 b의 성분 크기가 $b_1(R_1)>b_1(R_2)>b_1(R_3)$이므로 제품 1의 경우 $A_{R_1,J_{11}}=A_{R_2,J_{12}}=A_{R_3,J_{13}}=1$이고 다른 값은 0이 된다. 예를 들면 $A_{R_2,J_{11}}$의 경우 J_{11}이 R_1에서 수행되지만 $b_1(R_1)>b_1(R_2)$이므로 $A_{R_2,J_{11}}=0$이 된다. 제품 2에 대해서는 처리 공정이 반대 방향이 되어 두 번째 작업 단계의 경우 J_{21}이 R_3에서 수행되고 $b_1(R_2)>b_1(R_3)$이므로 $A_{R_2,J_{21}}=1$이 된다. 마찬가지 이유로 $A_{R_1,J_{21}}=A_{R_1,J_{22}}=1$이 되고, $A_{R_3,J_{21}}=A_{R_2,J_{22}}=A_{R_1,J_{23}}=1$이다. 두 번째 부등식도 같은 방법으로 설명될 수 있다.

그러나 m-RUN DAP의 성능은 선택된 순서 벡터 b와 합성된 단일 RUN DAP의 개수 m에 의해서 결정되며 다음 장에서는 주어진 m값에 대해 성능이 좋은 m개의 순서 벡터 $b_i(i=1,\cdots,m)$를 생성하는 방법에 대해 설명하고자 한다.

3. 시뮬레이티드 어닐링 기반 m-RUN DAP 생성 알고리즘

본 연구에서는 성능이 좋은 m-RUN DAP를 생성하기 위한 방법으로 하나의 m-RUN DAP를 찾은 후 지역 탐색 개념을 적용하는 메타 휴리스틱 방법 중의 하나인 시뮬레이티드 어닐링 알고리즘(Hillier 등, 2010)을 이용하여 개선된 m-RUN DAP를 찾고자 하였다.

3.1 초기 m-RUN DAP 생성 방법

r개의 워크스테이션으로 구성된 시스템에 대해 초기 m-RUN DAP는 Property 2에서 설명된 바와 같이 ${}_{r!}C_m$개가 존재하며 m개의 순서를 랜덤하게 생성하여 구할 수 있다. 이 때 본 연구에서는 m개의 순서 벡터 생성 시 중복은 허용하지 않으며, 2.2절에서 정의한 방법에 의해 초기 m-RUN DAP를 생성한다.

3.2 이웃 m-RUN DAP 생성

하나의 m-RUN DAP가 주어졌을 때 이웃 m-RUN DAP는 지역 탐색 기법의 개념을 다음과 같이 적용하여 생성하였다.: 먼저 m개의 순서 벡터$(b_1,\cdots,b_m)$에서 임의로 하나의 벡터 b_t를 선정하여 새로운 순서 벡터 b_t'으로 바꿔 새로운 순서 벡터를 만든다. 이 때 새로 생성된 순서 벡터는 기존 순서 벡터와 중복되지 않도록 $b_t' \neq b_i(i \neq t)$ 조건을 만족시킨다. 새로 생성된 순서 벡터에 대해 upstream neighborhood 행렬 A_t'을 생성하고 2.2절에서 정의한 방법에 의해 이웃 m-RUN DAP를 생성한다.

3.3 m-RUN DAP의 성능 평가

새로 생성된 m-RUN DAP의 성능은 찾을 수 있는 안전 상태 개수로 정의한다. 이 때 m-RUN DAP가 찾을 수 있는 안전 상태 집합은 m-RUN DAP를 구성하는 각각의 RUN DAP에 의한 안전 상태 집합의 합집합이 된다.

또한 현재의 m-RUN DAP와 이에 대한 이웃 m-RUN DAP의 성능 비교는 두 개의 DAP가 찾을 수 있는 안전 상태 집합의 크기로 평가한다. 만일 새로 선정된 이웃 m-RUN DAP가 더 낮은 성능을 가질 경우 이웃 m-RUN

DAP를 다시 선택할 수 있다. 그러나 본 연구에서는 해 영역(solution space)에 대한 효율적인 탐색(exploration)을 위해 개선되지 않는 방향으로의 검색도 일부 허용하는 시뮬레이티드 어닐링 방법을 적용하였다.

3.4 시뮬레이티드 어닐링 기반 m-RUN DAP 생성 알고리즘

그림 2는 제안된 시뮬레이티드 어닐링 기반 m-RUN DAP 생성 알고리즘의 순서도를 나타내며, 다음과 같이 세 단계로 구분된다.:

- 1단계: 초기화

초기화 단계에서는 여러 가지 파라메타를 초기화하고 초기 m-RUN DAP π_c를 생성하여 성능 값인 $N(\pi_c)$를 계산한다.

- 2단계: 반복 과정

반복 과정에서는 온도 스케줄에 따라 온도 T를 변화시켜가면서 π_c를 개선시킨다.

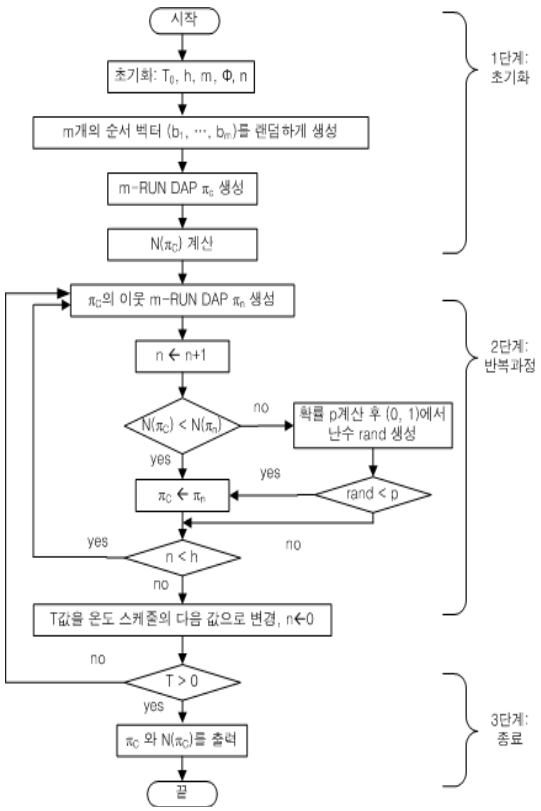

그림 2. 시뮬레이티드 어닐링 기반 m-RUN DAP 생성 알고리즘 순서도

- 3단계: 종료

계획된 모든 온도 변경 스케줄이 실행된 후에 알고리즘 수행을 종료한다.

이 때 그림 2에서 사용된 기호는 다음과 같다.:

- T_0 : 초기 온도
- h : 특정 온도 T에서 알고리즘의 반복 수
- Φ : 시스템 구성 정보(워크스테이션의 수, 버퍼 용량, 처리되는 job stages 수, job 처리 경로 등)
- π_c : 현재까지 찾은 가장 좋은 m-RUN DAP
- π_n : π_c의 이웃 m-RUN DAP
- $N(\pi)$: m-RUN DAP π가 찾을 수 있는 안전 상태의 수로서 $N(\pi) = \left|\cup_{i=1}^{m} S_i\right|$로 표현되며, 이 때 S_i는 순서 벡터 b_i에 의해 생성되는 RUN DAP가 찾을 수 있는 안전 상태의 집합을 나타냄($\pi = \pi_c$ 또는 π_n)
- p : 개선되지 않은 π_n을 선택할 확률로서 $p = \exp(\frac{N(\pi_n) - N(\pi_c)}{T})$로 계산됨

일반적으로 시뮬레이티드 어닐링 알고리즘에서 온도 스케줄 T는 일정한 비율에 의해 단계적으로 감소하도록 계획된다. 따라서 본 연구에서는 T의 감소율을 d라고 할 때 $T_i = d \bullet T_{i-1} (i = 1, 2, \cdots, 0 \le d \le 1)$ 으로 정의하고 $T_i > 1$를 만족하는 동안 알고리즘이 반복되도록 하였다(T의 변경 스케줄 정의에 의해서 T_i는 음수가 될 수 없기 때문에 알고리즘의 종료 기준 온도 값을 T가 1보다 작아졌을 때로 정함). 또한 제안된 알고리즘의 성능은 m 값이 주어졌을 때 특정한 온도 T에서의 반복 수 h와 온도 변화 스케줄에 의한 T의 감소율에 영향을 많이 받을 수 있다. 즉 h와 T의 감소율이 전체 알고리즘의 반복 수를 결정하는 주요 요인이 될 수 있으며, 알고리즘 성능에도 영향을 줄 수 있게 된다.

제안된 알고리즘의 기본적인 절차는 다음과 같다.: 알고리즘은 반복적으로 현재까지 찾은 가장 좋은 m-RUN DAP π_c에 대해 이웃 m-RUN DAP π_n을 생성하고 이들의 성능을 평가하여 개선 여부를 결정한다. 만일 π_n이 π_c보다 우수한 성능을 갖는다면(i.e., $N(\pi_c) < N(\pi_n)$) π_c를 π_n으로 대체한다. 그러나 그렇지 않은 경우라도 확률 p의 기회로 π_n을 새로운 해로 선택한다. 이 때 개선되지 않은 π_n을 현재까지 찾은 가장 좋은 m-RUN DAP로 선

표 1. 수치 실험을 위한 시스템 구성

	워크스테이션 수	버퍼 용량	job stages 수	job 경로
Conf 1	4	(2,1,4,2)	6	1→2→3→4→2→3
Conf 2		(6,5,3,4)		1→4→3→4→2→3
Conf 3	5	(2,3,5,1,3)	7	1→5→2→4→5→3→2

택할 확률 p의 크기는 온도 T의 값에 가장 큰 영향을 받는데, 온도 T는 알고리즘의 반복 수가 늘어날수록 작아지므로 이에 따라 확률 p는 점차 작아져 0에 가까워진다. 이러한 현상은 고려된 시뮬레이티드 어닐링 알고리즘이 반복 수가 늘어나면서 점차 우수한 해 영역으로 검색 방향이 수렴되면서 좋은 해 탐색에만 집중함을 의미한다.

4. 실험 설계 및 분석

4.1 실험 설계

제안된 m-RUN DAP 생성 방법의 성능을 평가하기 위해 본 연구에서는 몇 가지 샘플 시스템을 구성한 후 대상 시스템에 대한 수치적인 실험을 실시하였다. 표 1은 고려된 샘플 시스템의 구성을 나타낸다. 먼저 고려된 시스템은 워크스테이션이 4개와 5개인 경우로 구분하였으며, 각각의 경우에 대해서 버퍼 용량이 다른 시스템 구성을 생성하였다. 또한 한 가지 종류의 제품을 생산하는 것으로 가정하였으며, 시스템 내부에서의 재공품 흐름의 복잡성을 증가시키고 교착(deadlock) 현상을 발생시키기 위해 제품 처리 경로는 재진입(re-entrant)이 있는 경우를 고려하여 임의로 설정하였다. 이 때 RUN DAP 정의를 위한 행렬 A는 앞의 예에서와 같이 각 작업 단계에 대한 워크스테이션과 선행 작업들을 처리하는 워크스테이션에 대응되는 벡터 b값을 비교하여 정해진다.

각각의 시스템 구성에 대해 적용된 시뮬레이티드 어닐링 기반의 m-RUN DAP 생성 알고리즘의 성능 평가 결과는 개선되지 않은 이웃 m-RUN DAP π_n을 받아들일 것인지를 결정할 때 랜덤하게 생성되는 확률 값(rand)에 의해서도 영향을 받을 수도 있다. 따라서 본 연구에서는 동일한 시스템 구성에 대해서 제안된 m-RUN DAP 생성 알고리즘을 30번씩 반복 적용 한 후 각각의 최종 π_c에 대한 $N(\pi_c)$ 값의 평균을 대상 시스템에 대한 m-RUN DAP의 성능으로 평가하였다.

한편 m-RUN DAP의 성능은 선택된 m 값에 의해 결정될 수도 있기 때문에 각 시스템 구성에 대해 적정한 m 값을 찾기 위한 실험도 수행하였다. 이를 위해 r개의 워크스테이션으로 구성된 시스템에서 전체 가능한 순서 벡터 경우의 수 $r!$의 5%, 10%, 15%, 20%를 m값으로 선택하여 m-RUN DAP의 성능 변화를 관찰하였으며, 비교를 위해 100% 값에 대해서도 실험을 병행하였다.

표 2. 단일 RUN DAP 성능 실험 결과

		최적 DAP	단일 RUN	
			RUNc	RUNo
Conf 1	안전 상태 수	387	126	207
	% 커버리지	100	32.56	53.49
Conf 2	안전 상태 수	6174	2268	4403
	% 커버리지	100	36.73	71.32
Conf 3	안전 상태 수	3549	431	1794
	% 커버리지	100	12.14	50.55

4.2 실험 결과

고려된 DAP의 성능 평가를 위한 지표로 본 연구에서는 찾은 안전 상태 수와 % 커버리지를 이용하였다. % 커버리지는 전체 안전 상태 수에 대해 찾은 안전 상태 수의 비로 다음과 같이 나타낸다.:

$$\%\text{커버리지} = \frac{\text{찾은 안전상태수}}{\text{전체 안전상태수}}.$$

이를 바탕으로 수치 실험은 크게 두 가지로 구분되어 수행되었다.: (i) 단일 RUN DAP의 경우 버퍼 용량을 사용할 때와 순서 벡터를 사용할 때, (ii) m-RUN DAP.

먼저 표 2는 표 1에 있는 시스템 구성에 대해 단일 RUN DAP를 적용한 성능 실험 결과를 나타낸다. 실험 결과 단순히 버퍼 용량을 사용할 때(RUN_c) 보다 본 연구에서 제안된 알고리즘을 적용하여 m=1인 경우의 순서 벡터를 이용했을 때(RUN_o) 최대 35% 이상의 개선 효과를 확인할 수 있었다. 이 때 적용된 시뮬레이티드 어닐링 알고리즘은 $T_0 = 10{,}000, h = 20, d = 0.1$등의 파라메터 값

표 3. m-RUN DAP 성능 실험 결과

		m-RUN(m≥2)				
r!에 대한 사용된 m의 비율		5%	10%	15%	20%	100%
Conf 1	평균 반복 수	138	108	80	67	1
	평균 안전 상태 수	237	237	237	237	237
	평균 % 커버리지	61.24	61.24	61.24	61.24	61.24
Conf 2	평균 반복 수	41	34	32	33	1
	평균 안전 상태 수	4466	4466	4466	4466	4466
	평균 % 커버리지	72.34	72.34	72.34	72.34	72.34
Conf 3	평균 반복 수	173	133	191	176	1
	평균 안전 상태 수	2004	2008	1986	2000	2010
	평균 % 커버리지	56.47	56.58	55.96	56.35	56.64

을 사용하였으며 1,760번의 반복을 통해 충분히 수렴할 수 있도록 실행되었다.

한편 m-RUN DAP 알고리즘에 대한 실험 결과는 표 3과 같다. 두 번째 행은 전체 가능한 $r!$개의 순서 벡터 중에서 선택되어 사용된 순서 벡터 개수의 비율(%)을 나타낸다. 각각의 경우를 세 가지 시스템 구성에 대해서 30번의 실험을 실행하여 평균 반복 수, 평균 상태 수, 평균 % 커버리지를 계산하였다. 평균 반복 수는 100%, 즉 $r!$개의 순서 벡터를 사용하여 찾을 수 있는 최대 안전 상태 수만큼의 안전 상태를 찾는데 걸린 알고리즘의 평균 반복 수로서 30회 실행에 대한 평균값으로 계산하였다. 이 때 $r!$개의 순서 벡터를 사용하여 찾을 수 있는 최대 안전 상태 수는 표 3의 마지막 열에 표현되었다. 예를 들면, 5% 개의 순서 벡터로 구성된 m-RUN DAP를 Conf 1에 적용할 경우 237개의 안전 상태를 찾는 데 알고리즘을 30회 실행할 경우 실행 당 평균 138번의 반복 수가 소요되었다. 비교적 간단한 시스템 구성인 Conf 1과 Conf 2의 경우에는 적은 개수의 순서 벡터를 이용하면서도 최대 성능을 가지는 m-RUN DAP를 생성할 수 있었다(예를 들면 5%는 $m=2$, 10%는 $m=3$인 경우임). 또한 평균 % 커버리지도 단일 RUN DAP에 비해 매우 개선된 결과를 얻을 수 있었다.

그러나 Conf 3의 경우에는 전체 가능한 m의 개수가 120개이며 100% 모두 사용했을 경우 56.64%의 커버리지를 보였다. 이 경우 5% 또는 10%의 순서 벡터만을 사용하더라도 m의 개수가 각각 6개와 12개로서 매 반복마다 이 중에서 한 개의 순서 벡터를 랜덤하게 변경하기 때문에 전체적으로 알고리즘의 수렴 속도가 많이 떨어졌으며, 최대 안전 상태를 모두 찾기도 어려웠음을 발견할 수 있었다. 그러나 이 경우에도 $m=2$인 m-RUN DAP를 고려하였을 때 2010개의 안전 상태를 모두 찾는 경우도 실험을 통해 확인할 수 있었다.

4.3 분석 및 고려 사항

본 실험을 통해 단순히 버퍼 용량을 사용하는 것보다는 좋은 성능을 갖는 순서 벡터를 이용하는 것이 더욱 효과적인 것을 확인할 수 있었으며 그러한 순서 벡터는 본 연구에서 제안된 시뮬레이티드 어닐링 기반의 알고리즘을 이용하여 효율적으로 찾을 수 있다. 또한, 모든 시스템 구성에 대해서 주요한 특성을 갖는 순서 벡터들이 존재하며 최소한의 주요 특성 순서 벡터를 이용하여 최대한의 성능을 갖는 m-RUN DAP를 생성할 수 있을 것으로 사료된다.

그러나 제안된 알고리즘은 중간 단계에서 DAP의 성능을 평가하기 위해 $N(\pi_c)$ 값을 계산해야 되는데 본 실험에서는 시스템의 규모가 작기 때문에 모든 가능한 안전 상태를 열거하여 계산할 수 있었지만, 일반적으로는 이를 위한 좀 더 효율적인 근사화 방법이 필요하다. 또한 제안된 알고리즘의 성능은 선택된 초기 m-RUN과 이웃 m-RUN을 생성하기 위해 선택되는 순서 벡터에 영향을 많이 받기 때문에 이들을 효율적으로 선택하는 방법에 대한 개선도 필요하다.

5. 결 론

본 논문에서는 시뮬레이티드 어닐링 알고리즘에 기반한 다중 RUN(multi-RUN: m-RUN) 교착 제어 정책 생성 알고리즘에 대해 제안하였다. 이 방법은 단순한 RUN

DAP를 m개 생성한 후 이들의 합성에 의해 초기 m-RUN DAP를 정의하고 이를 점차적으로 개선시켜 나가는 것이다. 이 때 이웃 m-RUN은 현재 m-RUN에서 오직 한 개의 성분 RUN만을 랜덤하게 수정하여 생성하는 지역 탐색 기법을 적용하여 선택되었다.

또한 제안된 알고리즘의 성능 평가를 위해 몇 가지 기본적인 시스템 구성을 가정하고 수치 실험을 적용하였다. 그 결과 제안된 알고리즘은 버퍼 용량을 사용하는 단순 RUN DAP의 성능을 크게 개선시킬 수 있었으며, 대상 시스템에 대해서 모든 가능한 순서 벡터들 중에 m-RUN DAP의 성능에 크게 영향을 주는 최소한의 주요한 순서 벡터들이 존재하며 이들을 효율적으로 활용하는 것이 m-RUN DAP의 생성에 영향을 줄 수 있음을 실험적으로 확인하였다.

참 고 문 헌

1. Choi, J. Y. and Reveliotis, S. A. (2003), "A Generalized Stochastic Petri net Model for Performance Analysis and Control of Capacitated Re-entrant Lines", IEEE Trans. on Robotics & Automation, vol. 19, no. 3, pp. 474-480.
2. Gaarder, E. H. (1993), Deadlock Avoidance in Flexible Manufacturing Systems, Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL.
3. Hillier, F. S. and Lieberman G. J. (2010), Introduction to Operations Research, 9ed, McGrawHill.
4. Reveliotis, S. A. (2005), Real-time Management of Resource Allocation Systems: A Discrete Event Systems Approach, Springer, New York.
5. Reveliotis, S. A., Lawley, M. A., and Ferreira, P. M. (1997), "Polynomial comlexity deadlock avoidance policies for sequential resource allocation systems", IEEE Trans. on Automat. Contr., vol. 42, no. 10, pp. 1344-1357.
6. Revliotis, S. A., Lawley, M. A., and Ferreira, P. M. (2001), "Structural Control of Large-Scale Flexibly Automated Manufacturing Systems", in the Design of Manufacturing Systems, C. T. Leondes, Ed. Boca Raton, FL:CRC, pp. 4-1-4-34.
7. Reveliotis, S. A. and Ferreira, P. M. (1996), "Deadlock Avoidance Policies for Automated Manufacturing Cells", IEEE Trans. on robotics & Automation, vol. 12, no. 6, pp. 845-857.

최 진 영 (choijy@ajou.ac.kr)

1991 한양대학교 산업공학과 학사
1993 KAIST 산업공학과 석사
2004 Georgia Tech 산업및시스템공학과 공학박사
2007～현재 아주대학교 부교수

관심분야 : 스케쥴링, 모델링&시뮬레이션, 데이터 마이닝 최적화, 불확실성하의 의사결정