Roll out 알고리듬을 이용한 반복 작업을 하는 안전병렬기계 알고리듬 개발

- Development of an Algorithm for a Re-entrant Safety Parallel Machine Problem Using Roll out Algorithm -

백 종 관 *
Baek Jong Kwan
김 형 준 *
Kim Hyung Jun

Abstract

Among the semiconductor IC-chips, unlike memory chips, a majority of Application Specific IC(ASIC) products are produced by customer orders, and meeting the customer specified due date is a critical issue for the case. However, to the one who understands the nature of semiconductor manufacturing, it does not take much effort to realize the difficulty of meeting the given specific production due dates.

Due to its multi-layered feature of products, to be completed, a semiconductor product(called device) enters into the fabrication manufacturing process(FAB) repeatedly as many times as the number of the product specified layers, and fabrication processes of individual layers are composed with similar but not identical unit processes. The unit process called photo-lithography is the only process where every layer must pass through. This re-entrant feature of FAB makes predicting and planning of due date of an ordered batch of devices difficult.

Parallel machines problem in the photo process, which is bottleneck process, is solved with restricted roll out algorithm. Roll out algorithm is a method of solving the problem by embedding it within a dynamic programming framework. Restricted roll out algorithm is roll out algorithm that restricted alternative states to decrease the solving time and improve the result. Results of simulation test in condition as same as real FAB facilities show the effectiveness of the developed algorithm.

Keyword : Roll out Algorithm, Re-entrant Parallel Machine, FAB process

* 서일대학 산업시스템경영과 교수

1. 서 론

병렬 기계 환경은 많은 생산 현장에서 발생한다. 특히 최근 가장 치열한 경쟁 속에 있는 반도체(semiconductor)나 박막트랜지스터 액정표시장치(TFT-LCD ; Thin Film Transistor Liquid Crystal Display) 산업의 중요 공정(key process)인 FAB (FABrication)의 포토 공정(photo-lithography)이 병렬 기계로 구성되어있다. FAB 공장은 반도체 공정 중에서 가장 중요하면서 복잡하며 긴 공정 시간을 가진 공정으로, 웨이퍼 위에 반도체 회로를 새겨 넣어서 반도체 칩으로 만드는 과정이다. 웨이퍼 표면에 빛에 반응하는 광감응제(PR ; Photo Resist)를 반도체에 입히고 그 위에 원하는 패턴이 형성되어 있는 마스크를 통해서 빛을 쬐어주어 광화학 반응이 일어나게 한다. 그 다음에 반응한 물질을 현상(Develope)시켜 제거하면 원하는 모양의 패턴을 만들어 낼 수 있다. 식각을 통해 원하는 모양으로 깎아내고 남은 광감응제를 제거하면 한 층의 회로가 완성된다. 이런 과정을 여러 번 반복하여 웨이퍼 위에 반도체 칩을 생성하는 것이 FAB 공장이다. 실제로 하나의 제품이 생산되기 위해서, 반도체 FAB의 제조 과정은 포토 공정을 기준으로 보통 16 ~ 25 번의 반복적인 작업을 거치게 된다. 생산관리의 입장에서 보면 FAB 공장은 반복 작업이 가능한 병렬 공장으로 볼 수 있다. 병렬 작업장에서 같은 공정을 여러 번 반복적으로 수행하는 문제는 일반적인 접근으로는 해결할 수 없는 어려운 문제(NP-Hard)가 된다. 반도체 FAB 공정에서의 반복적인 작업은 병목 공정에서의 효과적인 생산 계획의 수립을 방해하는 요소가 된다. 복잡한 공정을 가진 문제에서는 병목 공정의 효율을 높여서 생산을 진행하는 것이 효과적인 방법이다. 반도체 FAB 공장에 대한 기존 연구들이 병목 공정 중심으로 소개되어 있고 Shifting Bottleneck 알고리듬과 같은 다른 연구에서도 병목 공정 중심으로 알고리듬이 개발되었다. 반도체 FAB 공장의 경우 반복적으로 병목 공정을 수행하기 때문에 병렬 기계로 이루어진 병목 공정을 효율적으로 운영하기 위한 병렬 기계 알고리듬의 적용을 어렵게 만드는 요소가 된다.

병렬 기계에 대한 기존 연구를 살펴보면 Randhawa 등[7]이 동일하지 않은 병렬 기계에서 생산 계획을 수립하기 위한 경험적 알고리듬(Heuristic Algorithm)을 개발하였다. Lee 등[4]은 병렬 기계에서 작업 순서에 따른 작업 준비 시간이 다른 경우에 대한 ATCS(Apparent Tardiness Cost with Setups)라는 알고리듬을 개발하였다. Hurink 등[2]은 작업 순서와 작업 준비시간이 존재하는 병렬 기계 문제에서 경험적 알고리듬을 통해 순서를 만들고 생산을 진행하는 방법을 리스트 생산 계획(List Scheduling)이라 하고 리스트 생산 계획의 영향과 장단점에 대해서 보였다. 병렬 기계에 대한 문제를 해결하기 위해 다른 접근 방법으로는 DP(Dynamic Programming)를 이용하여 접근한 Rothkopf[8]가 있으며 병렬 기계 중에서 가장 급한 기계에 작업을 투입하기 위한 작업의 순서를 정하는 방법으로 DP 해법을 사용하였다. Kim 등[3]은 DP 해법이 문제의 크기가 커짐에 따라 비약적으로 시간이 증가함에 따라 문제를 분해하여 작은 문제의 해를 DP 문제로 풀어서 전체 문제의 해로 접근해 가는 방법을 제안하였다. DP 해법의 유사한 형태로 Bertsekas 등[1]이 제안한 roll out 해법이 있다. 이는 최적해를 구하

는 방법이 아닌 경험적 알고리듬이나 DP 해법의 형태를 따르며 다른 경험적 알고리듬을 이용하여 알고리듬의 성능을 향상시키는 방법이다.

생산 관리의 목적도 빠르게 바뀌고 있다. 최근 세계적으로 시장이 단일화되어 가는 추세에서 모든 기업들은 글로벌 경쟁체제에 있다. 빠르게 돌아가는 현대의 경쟁적인 기업 환경의 특성을 비추어 볼 때 고객이 원하는 제품을 원하는 양만큼 원하는 시간에 공급해주는 것이 특히 중요하다. 과거 생산관리는 최소비용으로 최대한 많이 생산하는 것을 주로 연구하였으며 생산 관리의 효과를 평가하는 대표적인 지표도 출력률(throughput)이나 기계효율(utilization) 등이었다. 하지만 현대의 경쟁적인 조건에서 생산 관리의 목적은 고객의 만족도를 향상시키는 것으로 바뀌었다. 이런 목표 하에서 생산관리의 효과를 평가하는 지표는 주로 주문의 납기에 관련된 것이 대부분이다. 특히 반도체 제품과 같은 경우에는 전 세계가 하나의 시장으로, 가장 치열한 경쟁 속에 있는 제품 중의 하나이다. 또한 시설에 대한 투자비용이 크게 때문에 위험성이 큰 산업이다. 이러한 경쟁과 위험 속에서 살아남기 위해서는 품질 향상뿐만 아니라 고객의 요구를 만족시켜야 한다. 만약 고객 주문에 대한 납기 지연이 자주 발생한다면 고객에 대한 신용을 잃는 것뿐만 아니라 시장에서의 점유율도 잃게 될 것이다. 따라서 고객 만족이 생산 관리의 1차 목표가 되며 이에 따라 주문의 납기 만족이 생산 관리 효과를 나타내는 지표가 된다.

반도체 FAB 공장은 병렬 기계들로 이루어진 공정이며 같은 공정을 여러 번 반복적으로 수행하는 복잡한 공정을 가졌기 때문에 효과적인 통제 정책을 개발하는 것은 어렵다. 이는 기존의 연구들이 산출량 중심의 투입 정책에만 연구를 집중한 이유이기도 하다. 특히 반도체 FAB 공장이 병목 공정에 대한 반복 작업을 수행하는 공정 흐름을 가졌기 때문에 납기를 만족하기 위한 병목 공정에서의 효과적인 통제 정책 개발은 더욱 어렵다. 따라서 본 연구에서는 병목 공정의 효율을 높이기 위하여 병렬 기계 알고리듬을 개발하였다. roll out 알고리듬을 이용하여 병렬 기계 알고리듬을 개발하였으며 개발된 알고리듬을 적용하여 평균 납기 지연 시간을 줄이기 위한 병목 공정의 세부 생산 계획을 수립한다.

2. 본 론

2.1 병렬 기계 문제의 정의

반도체 공장 FAB 공정의 효율적인 운영을 위해, 병목 공정인 포토 공정에서 병렬 기계의 세부적인 생산 계획을 통해 병목 공정의 효율성을 높이기 위한 생산 계획을 수립한다. FAB 공정의 병렬 기계의 수는 포토 기계의 수가 되고 작업의 수는 제품을 만들기 위해 반복해야 하는 작업의 수와 같다.

병렬 기계 알고리듬을 어렵게 하는 다른 요소는 마스크의 존재이다. 병목 공정인 포토 공정에서 작업을 하기 위해서는 마스크(Mask)라는 장비가 필요하다. 마스크는 설계된 회로가 그려진 장비로써 포토 공정에서 마스크를 이용해 웨이퍼에 회로를 그려

넣게 된다. 마스크는 한 종류의 제품에서 한 회로층 당 하나씩 존재하는 것이 기본이며 따라서 포토 공정이 병렬 작업을 하지만 한꺼번에 모든 작업을 진행할 수 있는 것이 아니라 마스크의 이용 가능성에 따라 기계가 휴식 중이고 대기하는 작업이 존재하더라도 할 수 없다. 이런 마스크의 존재는 병렬 기계 문제를 한층 더 어렵게 만드는 요소가 된다. 따라서 포토 공정에서는 작업 준비 시간이 존재하는데 작업해야할 마스크가 기계에 미리 장착되어있는 경우에는 작업 준비 시간은 아주 작기 때문에 무시할 수 있다. 하지만 작업해야 할 마스크가 기계에 미리 준비되어 있지 않을 경우에는 긴 작업 준비 시간을 가지게 된다. 실제 공장에서 기계에 미리 장착할 수 있는 마스크의 수는 보통 5 개 정도이다. 본 연구에서의 기본적인 가정은 다음과 같다.

1. 각 기계는 하나의 작업에 대하여 가공이 완료될 때까지 다른 작업을 할 수 없다. (Nonpreemptive)
2. 기계에 장비가 미리 준비되지 않은 작업에 대해서는 작업 준비 시간이 존재한다.
3. 작업 준비 시간은 생산 순서에 관계없이 정해져 있다.
4. 하나의 기계에서 미리 준비할 수 있는 마스크의 수는 5개이다.
5. 각 작업의 공정에 대한 마스크는 하나이며 병렬 기계가 쉬고 있더라도 같은 장비를 사용하는 작업은 한 대의 기계에서만 진행할 수 있다.

문제를 정의하면, 작업의 시작 가능 시점과 납기가 존재하고 작업 간의 선행관계와 작업 준비 시간이 존재하는 병렬 기계에서 생산 계획을 수립하는 문제이다. 작업을 위해 필요한 마스크가 기계에 미리 할당되어 있지 않을 경우에만 생산 준비 시간이 필요하며 작업의 순서와는 상관없이 일정하다. 하나의 기계에는 5개의 마스크를 할당해 둘 수 있다. 이 병렬 기계 문제의 목적 함수는 납기를 어기지 않도록 하는 것이다. 즉, 최종 공정에서의 납기 준수가 목적이며 납기를 어긴 시간의 평균을 최소화하는 것이다. 따라서 다음과 같이 병렬 기계 문제를 정의할 수 있다.

목적 함수 : $G = \sum_{i=1}^{N} T_i / N$

(또는 $G = \sum_{i=1}^{N} T_i$

표기법(Notation)

i : 작업 번호 ($i = 1,2,3,\ldots$), N

j : 공정 번호 ($j = 1,2,3,\ldots$), n_i

k : 기계 번호 ($k = 1,2,3,\ldots$) M

p_{ij} : i 작업의 j 공정의 가공 시간

P_i : i 작업의 총 가공 시간 ($P_i = \sum_{j=1}^{p} p_{ij}$)

n_i : i 작업의 총 공정 수

R_i : i 작업 전체 공정의 시작 가능 시점

r_i : i 작업 현재 공정의 시작 가능 시점

D_i : i 작업의 전체 공정의 납기

s : 생산 준비 시간 (마스크 교체 시간)

C_i : i 작업의 종료 시점

T_i : i 작업의 납기를 어기는 시간 ($T_i = \max(0, C_i - D_i)$)

S_{kt} : t 시점에 k 기계에 걸려있는 마스크의 집합

i 작업의 j 공정들 ($S_{kt} = \{ (i,j), \ldots)\}$

RM_k : k 기계의 작업 가능 시점

$RMask_{ij}$: i 작업의 j 공정에 대한 마스크의 작업 가능 시점

$I_{ij}(k, t) = \begin{cases} 1 & \text{if } (i,j) \in S_{kt} \\ 0 & o/w \end{cases}$: i 작업의 j 공정에 대한 마스크가 t 시점에서 k 기계에 걸려있으면 1, 없으면 0

문제를 정의하면 목적 함수는 최종 납기를 어기는 납기 지연 시간의 합을 최소화하는 것이다. M 대의 동일한 병렬 기계로 이루어져 있으며 N 개의 작업이 존재한다. 각 작업은 작업 가능 시점(Release Time)과 납기(Due Date)를 가지고 있으며 작업간의 선행 관계도 존재한다. 주어진 문제는 직접적인 방법으로 접근할 수 없는 어려운 문제(NP-Hard)가 된다. 따라서 주어진 병렬 기계 문제를 효과적으로 해결할 수 있는 알고리듬을 개발하였다.

2.2 roll out 알고리듬

본 연구에서는 앞에서 언급한 선행 관계 및 작업 준비 시간과 작업 가능 시간, 납기가 존재하는 병렬 기계 문제를 풀기 위하여 roll out 알고리듬을 적용한다. 주어진 문제는 수학적으로 풀기 어려우며 효과적인 경험적 알고리듬(Heuristic Algorithm)조차 찾아보기 힘들다. 일반적으로 복잡한 문제에 대한 경험적 해를 구하기 위하여 리스트 생산 계획(List Scheduling)이라는 방법이 많이 이용된다. 리스트 생산 계획이란 생산 계획 대상이 되는 작업을 간단한 분배 정책 등을 이용하여 우선 순위를 정하고 리스트화하여 우선 순위가 높은 작업부터 차례로 투입하는 정책이다. 리스트 생산 계획은 복잡한 공정이 아닌 간단한 공정, 즉 한 대의 기계를 가진 문제(Single Machine Problem)에서 주로 사용되며 병렬 기계로 확장되어 적용된다. 리스트 생산 계획은 한 대의 기계를 가진 문제나 이 문제가

확장된 병렬 기계 문제와 같은 간단한 환경에서는 문제의 조건에 따라 최적 해를 찾는 알고리듬이 되는 경우도 있으나 대부분의 경우 경험적 알고리듬이 되며 짧은 생산 계획 시간 안에 좋은 가능 해를 찾는다는 장점이 있다.

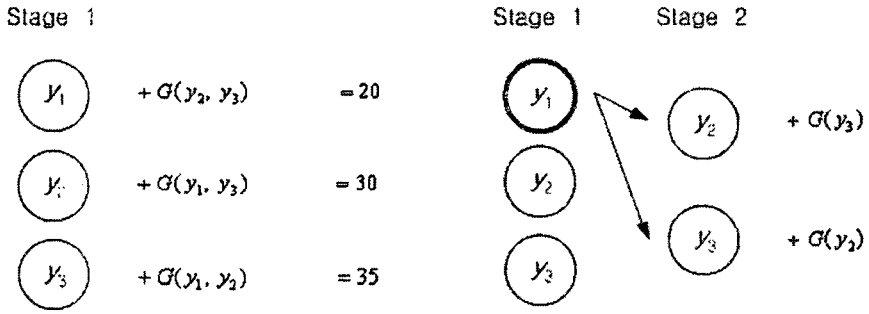
리스트 생산 계획에 이용되는 정책으로는 FIFO(First In First Out), EDD(Earliest Due Date), MS(Minimum Slack), CR(Critical Ratio), SPT(Shortest Processing Time), WSPT(Weighted Shortest Processing Time), SRPT(Shortest Remaining Processing Time), S/RPT(Slack per Remaining Processing Time), COVERT(Cost OVER Time), ATC(Apparent Tardiness Cost), ATCS (Apparent Tardiness Cost with Setups) 등이 있다. 일반적으로 FIFO와 WSPT, SPT, SRPT 등의 정책들이 많이 사용되며 납기를 만족시키기 위한 공장에서는 EDD 정책이 많이 사용된다. 또한 MS, COVERT, ATC, ATCS, S/RPT 등의 정책도 납기만족을 위해 사용될 수 있는 정책들이다. 복잡한 공정 과정을 가진 반도체 FAB 공장에서도 FIFO나 MS같은 간단한 리스트 생산 계획을 이용하여 생산을 진행한다. 한 대의 기계 환경에서는 특수한 몇 가지 조건을 제외하고는 ATC-COVERT-WSPT-SRPT-EDD-FIFO 등의 순서로 좋은 결과를 산출한다고 알려져 있다.[5]

문제가 복잡해지면 최적 해를 구하기 위한 알고리듬들은 시간상의 제약에 의해 직접 적용할 수 없기 때문에 빠른 시간에 해를 구할 수 있는 리스트 생산 계획이 많이 이용된다. 하지만 리스트 생산 계획은 단순한 문제에서는 효과적인 해를 제공하지만 복잡한 문제에 대해서는 좋은 해를 제공하지 못한다. 이러한 리스트 생산 계획의 단점을 보완하기 위해서 본 연구에서는 roll out 알고리듬을 이용한다.

roll out 알고리듬이란 DP(Dynamic Programming) 해법을 기본으로 여러 가지 형태의 조합 최적화(combinatorial optimization) 문제에 대한 개략적인 해를 제공할 수 있는 알고리듬이다[1]. DP 해법은 최적해(optimal solution)를 찾을 수 있지만 해를 구하기 위해서는 많은 시간과 큰 메모리 공간이 소요되는 문제점을 지닌다. 특히 문제의 크기가 커질수록 소요되는 계산 시간이 기하급수적으로 커지기 때문에 복잡한 형태를 지니며 빠른 계산 시간을 요구하는 현실 문제에는 적용될 수 없다는 단점을 지닌다. 따라서 유효한 계산 시간 안에 적절한 해를 구하기 위해서 이런 DP 해법을 응용한 roll out 알고리듬이 사용될 수 있다. roll out 알고리듬은 DP 해법과는 달리 각 단계(stage)에서 경험적(heuristic) 알고리듬을 통해 가장 좋은 해를 제공할 가능성이 있는 하나의 상태(state)만을 선택한 후 다른 상태들은 버리고 다음 단계로 전진한다. 따라서 방대한 계산 시간을 요구함으로써 유효 시간 안에 해를 찾을 수 없는 DP 해법에 비해 roll out 알고리듬은 유효한 계산 시간 안에 적절한 해를 구할 수 있다는 장점을 지닌다.

예를 들어 < 그림 1 >과 같이, y_1, y_2, y_3라는 3개의 변수를 순서대로 결정해야 하는 문제에서 첫 번째 단계(stage)에서는 3개의 변수 y_1, y_2, y_3이 가능 대안이 될 수 있다. roll out 알고리듬은 각각의 대안이 선택되어졌다고 가정하고 최종 해에 대한 목적함수의 값을 계산한다. 먼저 y_1이 결정되었다고 가정하고 남아있는 두개의 변수 (y_2, y_3)를 대상으로 경험적 알고리듬을 이용하여 해를 구한다. 마찬가지 방법으로

y_2, y_3이 결정되었다고 가정한 후 목적함수의 값을 계산한다. 각각의 목적함수의 값이 20, 30, 35가 나왔다면 가장 좋은 목적함수 값을 가진 변수 y_1을 첫 번째 단계의 해로 결정한다. 같은 방법으로 y_1이 결정된 상태에서 두 번째 단계의 해를 결정하기 위해 같은 방법으로 진행한다. 즉, 첫 번째 단계에서 y_1이 결정된 상태에서 두 번째 단계에서의 해를 결정하기 위해 두 번째 단계에서 y_2, y_3이 결정되었다고 가정한 후에 각각의 최종 목적함수 값을 계산한 후 작은 목적함수 값을 만드는 해를 결정한다. 이와 같은 방법으로 각 단계에서의 모든 대안에 대해 최종 목적함수의 값을 예측한 후에 가장 좋은 대안을 결정하고 다음 단계로 진행한다.



< 그림 1 > roll out algorithm.

2.3 비용 추정 함수 알고리듬

roll out 알고리듬의 해는 $G(f)$를 추정하기 위한 경험적 알고리듬에 의해서 결정된다. 기존의 roll out 알고리듬은 앞에서 언급한 기존 리스트 생산 계획 알고리듬을 그대로 사용할 수 있다. 하지만 $G(f)$ 추정 알고리듬의 속도와 효과가 roll out 알고리듬에 직접적인 영향을 미친다. 따라서 본 연구에서는 주어진 병렬 기계 문제를 풀기 위하여 앞에서 제안된 리스트 생산 계획 알고리듬보다 효과적인 비용 추정 함수를 개발하였다.

최종 작업의 납기를 지키기 위해서 기계에 대한 작업 준비 시간과 작업의 여유 시간을 동시에 고려한다. 가장 빨리 작업할 수 있는 기계에서 작업 준비 시간까지 고려한 가장 급한 작업을 선택하고 가장 급한 작업이 납기를 어기지 않는 범위에서 가장 급한 작업보다 먼저 진행할 수 있는 작업이 존재하는 지 검사하여 존재하면 그 작업을, 존재하지 않으면 가장 급한 작업을 진행하여 납기를 어기는 작업을 만들지 않으면서 기계의 효율을 높일 수 있는 작업을 투입한다. 따라서 본 연구에서는 기계의 효율을 높이면서도 납기를 어기지 않도록 하는 경험적 알고리듬을 개발하였다. 알고리듬을 정리하면 다음과 같다.

Step 0.

$A \leftarrow \{(1,1), (2,1), \ldots$ 해야 할 작업군 A에 각 작업이 진행해야 할 공정을 넣는다.

Step 1.

$K \leftarrow \min_k RM_k$ ($k = 1, 2, \ldots,$)M 가장 먼저 작업을 시작할 수 있는 기계 K를 찾는다.

$t \leftarrow RM_K$: 현재 시점을 선택된 기계의 가공 시작 시점으로 둔다.

Step 2.

모든 A에 대하여

$\min_A (D_i - \sum_{l=n_i}^{j} p_{il} - R)$ A_{MST}를 찾고

$\min_A \{ \max(r_i, RMask_{ij}, RM_K) + s \times I_{ij}(K, t) + p_{ij}$인 A_{MFT},

$\min_A \{ \max(r_i, RMask_{ij}) \}$인 A_{MRT}을 찾는다.

Step 3.

만약 $FT_{A_{MFT}} \leq \{ \max(r_i, RMask_{ij}, RM_K) \}_{A_{MST}}$ 이면 A_{MFT} 공정을 K 기계에 투입한다.

만약 $FT_{A_{MRT}} \leq \{ \max(r_i, RMask_{ij}, RM_K) \}_{A_{MST}}$ 이거나

$\left\{ D_i - s \times I_{ij}(K, FT_{A_{MRT}}) - \sum_{l=n_i}^{j} p_{il} \right\}_{A_{MST}} - FT_{A_{MRT}} \geq 0$이면 A_{MRT} 공정을 K 기계에 투입하고 아니면 A_{MST} 공정을 K 기계에 투입

$(FT_{A_{MFT}} = \{ \max(r_i, RMask_{ij}, RM_K) + s \times I_{ij}(K, t) + p_{ij} \})_{A_{MFT}}$

Step 4.

$RM_K \leftarrow \{ \max(r_i, s \times I_{ij}(K, t) + RMask_{ij}, RM_K) + p_{ij} \}_{A_{selected}}$

$r_{selected} \leftarrow \{ \max(r_i, s \times I_{ij}(K, t) + RMask_{ij}, RM_K) + p_{ij} \}_{A_{selected}}$

Step 5.

만약 $A_{Selected}$ 가 마지막 공정이면 A에서 제외

마지막 공정이 아니라면

$A \leftarrow A - A_{selected}$

$A \leftarrow A \cup A^{+}_{selected}$

단 $A^{+}_{selected}$ 는 $A_{selected}$ 작업의 다음 공정

Step 6.
만약 $A = \emptyset$이면 계획 끝
아니면 Step 1. 로

2.4 제한된 roll out 알고리듬

roll out 알고리듬이 DP 해법에 비해 빠른 계산 속도를 가지는 것은 $G^*(J)$를 계산하는 시간의 단축뿐만 아니라 각 단계(stage)에서 모든 상태(state)를 검사하지 않고 실제 좋아질 수 있는 대안(state)만 검사하여 불필요한 검사를 줄일 수 있기 때문이다. 대안의 크기를 줄이는 방법은 branch and bound 알고리듬의 fathoming 방법과 유사한 과정으로 생각할 수 있다. 따라서 얼마나 효과적으로 대안을 선택하고 줄일 수 있느냐가 수행 속도와 결과에 큰 영향을 미친다. 본 연구에서 대안을 제한하여 대안의 수를 줄여서 속도를 높이는 방법을 제한된 roll out(restricted roll out)이라고 한다. RRO(Restricted Roll Out) 알고리듬은 대안을 줄여서 속도를 빠르게 할 뿐만 아니라 roll out 알고리듬이 부분 최적해(Local Optimal Solution)에 빠지는 것을 미리 막아주어 해의 결과를 더 좋게 만들 수 있다.

대안을 선택하기 위해서 먼저 같은 종류의 작업에서 같은 공정에서는 하나의 작업만이 대안이 된다. 즉, 같은 마스크를 사용해야 하는 작업은 가장 급한 작업 하나만을 대안으로 한다. 다른 마스크를 사용하는 공정끼리는 좋은 해를 만들기 위한 대안이 되지만 같은 마스크를 사용하는 공정끼리는 더 급한 작업이 투입되는 것이 좋은 해를 만든다. 선택된 대안 작업에 대하여 대안 기계를 선택하기 위하여 먼저 대안 작업의 공정에 대한 마스크가 할당되어 있는 기계는 언제나 대안 기계가 된다. 또한 가장 먼저 쉬게 되는 기계도 대안 기계가 된다. 여기에 공정이 시작할 수 있는 시점보다 할당 가능 시점이 늦은 기계도 대안 작업에 대한 대안 기계가 된다. 이런 방법으로 roll out에서 비용 함수를 계산해야 하는 대안을 줄여서 속도를 높여주고 또한 roll out이 투입 가능한 대안만을 선택해줌으로써 부분 최적해에 빠지는 것을 막아서 해를 좋게 할 수 있다. RRO 알고리듬에서 대안의 선택을 제한하는 방법은 다음과 같다.

Step 0.
$A \leftarrow \{(1,1), (2,1), \ldots\}$대안 작업군 A에 각 작업이 진행해야 할 공정을 넣는다.
$L \leftarrow |A|$, $l \leftarrow 1$, $S \leftarrow \emptyset$

Step 1.
$K \leftarrow \min_k RM_k$ ($k = 1, 2, \ldots,$)M 가장 먼저 작업을 시작할 수 있는 기계 K를 찾는다.
$t \leftarrow RM_K$: 현재 시점을 선택된 기계의 가공 시작 시점으로 둔다.
A를 max (r_i, $RMask_{ij}$)의 값이 작은 순으로 정렬한다.

Step 2.

A_l 선택 (A_l 은 A에서 정렬된 l 번째 작업 공정)

Step 3.

만약 $A_l \notin S$ 이면 Step 4. 로

만약 $A_l \in S$ 이면 Step 5. 로

Step 4.

$\{ \max(r_i, RMask_{ij}) \}_{A_l} \leq RM_k$ 이거나 $I_{A_l}(k,t) = 1$인 k가 있다면 만족하는 모든 k에 대하여 $S \leftarrow S \cup \{(A_l, k)\}$

$S \leftarrow S \cup \{(A_l, K)\}$

Step 6.으로

Step 5.

$A_l \simeq S_{same}$ 인 모든 S_{same} 에 대하여 ($S_{same} \in S$)

만약 $(D_i - \sum_{l=n_i}^{j} p_{il} - t)_{A_l} \leq (D_i - \sum_{l=n_i}^{j} p_{il} - t)_{S_{same}}$ 이고

$FT_{A_l} > \{ \max(r_i, RMask_{ij}, RM_{A_l}) \}_{S_{same}}$ 이고

$\left\{ D_i - \sum_{l=n_i}^{j} p_{il} \right\}_{A_l} - FT_{S_{same}} \leq \theta$ 이면 $S_{same} \leftarrow A_l$

(단, $A_l \simeq S_{same}$ 은 S_{same} 의 작업 공정 부분이 A_l 과 같음

$FT_{S_{same}} = \{ \max(r_i, RMask_{ij}, RM_k) + s \times I_{ij}(k,t) + p_{ij} \}_{S_{same}}$)

Step 6.

만약 $l = L$이면 대안 선택 끝

아니면 $l \leftarrow l + 1$ Step 1. 로

2.5 결과 분석

roll out 알고리듬은 FAB 공장이 포토 공정만 존재하는 병렬 기계라고 가정하고 알고리듬의 효과를 분석하였다. 병렬 기계의 수와 작업의 종류는 처음 제안한 FAB 공장의 기본 구성을 따랐으나 실험 시간을 단축하기 위하여 병렬 기계의 수는 기본 구성보다 적게 구성하였다. 기계의 수를 적게 한 것은 RRO 알고리듬과 roll out 알고리듬, 그리고 비용 함수 계산 알고리듬의 효율성을 비교하는 것에는 영향을 미치지 않는다.

실험을 위해 주어진 문제는 병렬 기계만 존재하며 병렬 기계를 N 번 반복적으로 거쳐야만 작업이 끝난다. 병렬 기계와 병렬 기계 사이에는 반드시 쉬어야하는 시간이 존재하여 한번 병렬 기계를 거친 후에는 일정 시간을 거친 후에 다시 가공될 수 있다. 이것은 반도체 FAB 공장에서 포토 이외 공정의 생산 시간을 위한 가정으로 작업 후 다음 작업을 위한 이동시간으로 가정한다.

본 연구에서 제안된 비용 함수 계산 알고리듬과 비교하기 위하여 총 6가지의 리스트 생산 계획 알고리듬과 비교 실험하였다. 병렬 기계에서 납기 만족을 위해 적용할 수 있는 MS, ATC, ATCS를 실험하였고 각 알고리듬에서 중요도라는 개념을 포함한 WMS, WATC, WATCS 등도 실험하였다. 여기서 중요도는 얼마나 뒤쪽 공정인지를 나타내는 것으로 실제 각 작업은 같은 중요도를 가진다. 따라서 중요도(W)는 $W = j/n$로 계산할 수 있으며 최종 공정으로 갈수록 1에 가까워지면서 커진다. 실험된 WMS(Weighted Minimum Slack)은 본 연구에서 실험을 위해 제안한 알고리듬으로 여유 시간(Slack)에 중요도를 나누어 우선 순위를 주었다. 같은 여유 시간을 가지더라도 뒤쪽 공정으로 갈수록 여유 시간이 커지게 된다.

가공 준비 시간에 대한 대안으로는 전혀 가공 시간이 필요 없는 대안에서부터 병렬 기계 가공 시간과 동일한 가공 준비 시간이 필요한 대안, 가공 준비 시간이 병렬 기계의 두 배인 대안으로 실험하였다.

각 대안에 대하여 roll out을 사용하지 않고 비용 함수 계산을 위해 제안된 알고리듬으로 생산 계획을 세우는 방법과 대안을 제한하여 알고리듬의 속도를 높인 RRO 알고리듬을 이용한 방법, 그리고 roll out 알고리듬을 이용한 방법으로 세 가지 대안에 대하여 실험하였다. 모의 실험에서 대안은 다음과 같다.

- 병렬 기계 수(MP) : 3, 6, 9
- 작업의 종류(N) : 5, 10, 15
- 작업의 수 : 100, 200, 300
- 병렬 기계에서의 반복 횟수(SL) : $U[3,5]$ $U[5,10]$ $U[8,15]$
- 포토 공정의 생산 시간 : 100
- 병렬 기계 작업 후 다시 병렬 기계까지의 시간(p_j) : $U[200,800]$
- 마스크 교체 시간 : 0, 100, 200
- 작업 가능 시점 : $\exp(\lambda)$

  주문 발생의 시간 간격(Inter Arrival Time)을 λ 라 두면

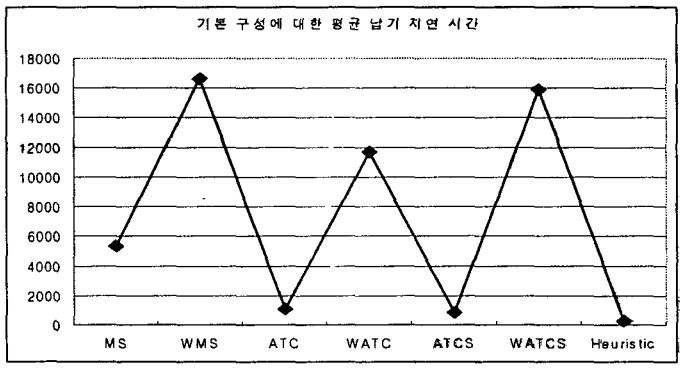
  $$\lambda = \left(\sum_{i=1}^{N} 100 \times SL_i / N \right) / MP / R$$

  R : 0.8, 1.0, 1.2
- 주문의 납기(D_i) : $R_i + MOT * SL_i$

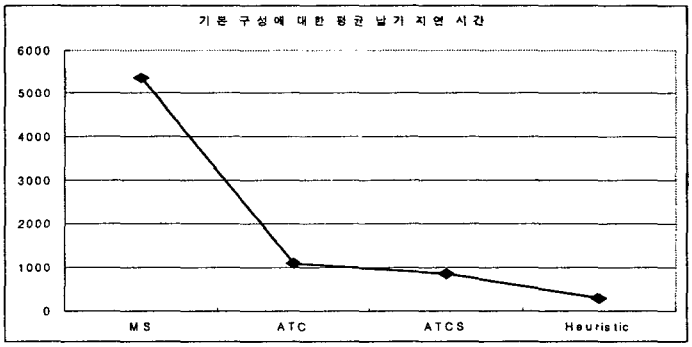
  평균 공정 시간(Mean Operation Time)을 MOT 라 두면

$$MOT = 4.0 \times \left\{ \sum_{i}^{N} \left(\sum_{j}^{SL_i} p_j / SL_i \right) + 100 \right\}$$

- 총 대안 수 : 17(문제 구성 대안) × 7(비용 함수 대안) × 3(*Roll Out* 대안) = 357
- 대안에 대한 실험 횟수 : 10
- 총 실험 횟수 : 357(총대안수) × 10(대안당실험횟수) = 3570

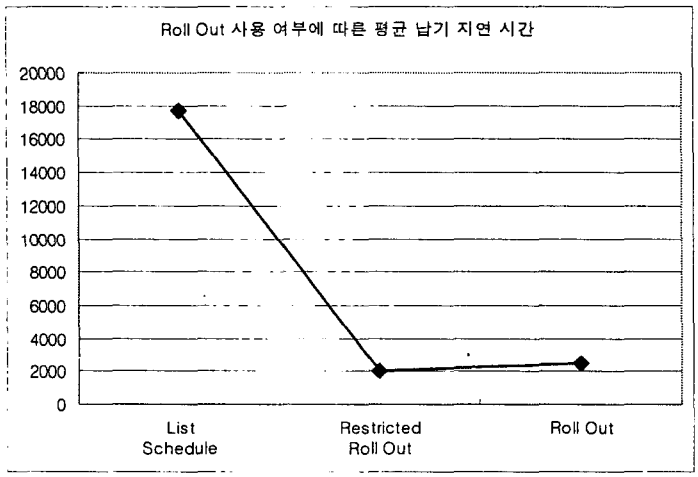


< 그림 2 > 기본 구성에 대한 리스트 생산 계획의 비교

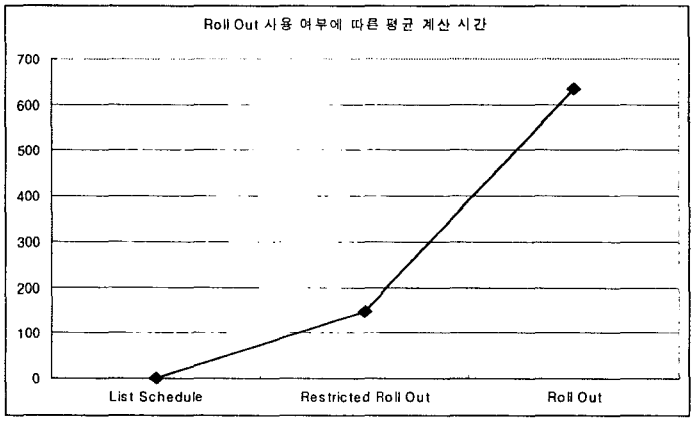


< 그림 3 > 기본 구성에 대한 리스트 생산 계획의 비교 - 2

< 그림 2 >에서 보면 리스트 생산 계획들 중에서 공정에 대한 가중치(Weight)를 준 정책이 더 안 좋은 결과를 가져오는 것을 알 수 있다. 작업들은 뒤쪽 공정으로 갈수록 더 급한 작업이 되어, 앞쪽 공정을 진행 중이지만 실제로는 더 급한 작업들을 뒤로 밀리게 된다. 따라서 공정에 대한 가중치를 부여하여 최종 납기에 가까운 공정일수록 인위적으로 더 급한 작업으로 두는 것은 가중치가 없는 대안보다 더 나쁜 결과를 산출하게 만든다. < 그림 3 >에서처럼 가중치를 부여하지 않은 알고리듬만을 비교하면 제안한 경험적 알고리듬을 사용하는 것이 가장 효과적인 것으로 나타났다.



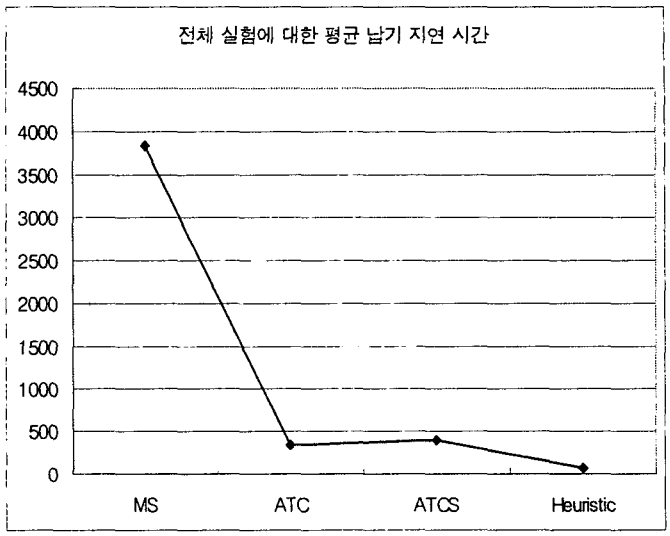
< 그림 4 > 기본 공정에서 roll out 사용 여부에 따른 평균 납기 지연



< 그림 5 > 기본 공정에서 roll out 사용 여부에 따른 평균 계산 시간

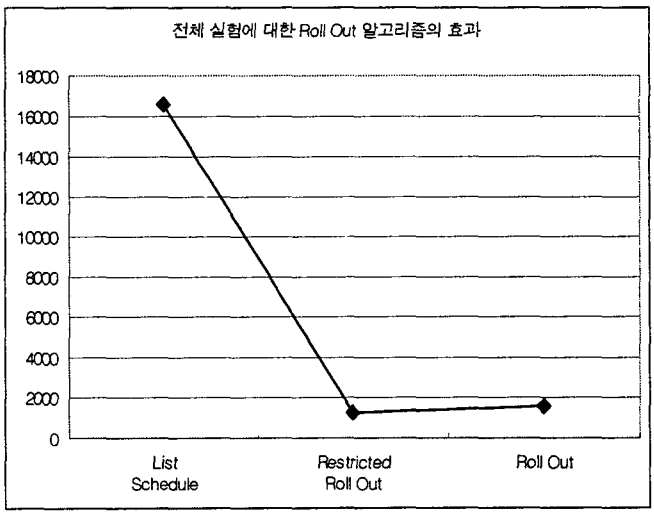
< 그림 4 >을 보면 roll out 알고리듬보다 본 연구에서 제안한 RRO(Restricted roll out) 알고리듬이 평균 납기 지연 시간에 대하여 더 좋은 결과를 나타내었다. roll out 알고리듬도 DP 형태를 빌려온 일종의 경험적 알고리듬이라고 할 수 있으며 부분 최적해(Local Optimal)에 빠질 위험을 가지고 있다. 본 연구에서 제안한 RRO 알고리듬은 roll out 알고리듬이 부분 최적해에 빠지지 않도록 나쁜 대안을 제거해주기 때문에 오히려 더 좋은 결과를 가져왔다. 그리고 또한 < 그림 5 >에서 해를 구하기 위한 계산 시간에서 리스트 생산 계획만으로 해를 구한 시간은 평균 1초도 걸리지 않았다. 비용 함수를 추정하기 위해 RRO 알고리듬과 roll out 알고리듬은 리스트 생산 계획을 그 일부로 사용하였기 때문에

더 많은 계산 시간이 필요하였으며 RRO 알고리듬이 roll out 알고리듬보다 대안의 수를 줄여서 계산하기 때문에 계산 시간에서 많은 차이를 나타냈다.

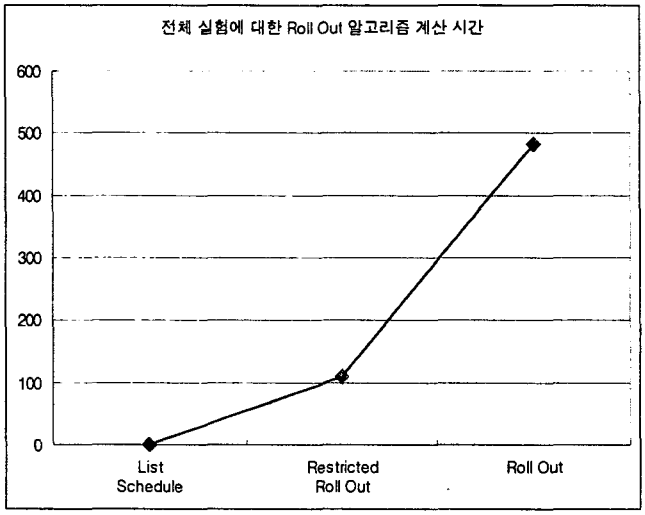


< 그림 6 > 전체 실험에 대한 평균 납기 지연 시간

< 그림 6 >에서 기본 구성이 아닌 전체 실험에 대한 결과에서도 기본 구성과 같은 결과를 나타냈다. 공정에 대한 가중치를 부여하는 알고리듬도 똑같은 패턴을 나타내었으며 그림에서는 제외하였다. 제안된 경험적 알고리듬이 다른 알고리듬에 대하여 평균적으로 좋은 결과를 보였다.



< 그림 7 > 전체 실험에 대한 roll out 알고리듬의 효과



< 그림 8 > 전체 실험에 대한 roll out 알고리듬 계산 시간

< 그림 7 >와 < 그림 8 >에서 roll out 알고리듬의 사용에 따른 결과를 보면 기본 공정에서의 결과와 같은 형태를 보여준다. RRO 알고리듬을 사용하는 것이 가장 좋은 결과를 보였다. roll out 알고리듬을 사용한 결과보다 좋은 해를 구했으며 리스트 생산 계획에 비해서는 상당한 차이를 보였다. 계산 시간에서도 roll out 알고리듬에 비해 상당히 작은 시간을 요구했다.

3. 결 론

본 연구에서는 병목 공정인 병렬 기계 문제에서의 효율적인 해를 얻기 위해 roll out 알고리듬을 소개하고 적용하였다. roll out 알고리듬의 비용 함수로 사용할 수 있는 효과적인 알고리듬을 제안하였으며 RRO(Restricted roll out) 알고리듬을 개발하여 실제 문제에 적용이 가능하도록 roll out 알고리듬의 속도와 결과를 향상시켰다. RRO 알고리듬은 roll out 알고리듬의 대안을 줄여서 부분 최적해에 빠지지 않도록 함으로써 속도와 결과를 향상시키는 새로운 방법이다. 모의 실험에서 제안된 경험적 알고리듬을 비용 함수로 사용하는 것이 다른 리스트 알고리듬을 사용하는 것보다 상당히 좋은 결과를 얻었으며 제안한 RRO 알고리듬은 수행 속도와 결과 면에서 roll out 알고리듬보다 좋은 값을 보여줬다.

추후 연구 과제로는 병렬 기계 알고리듬의 효율을 높이기 위하여 비병목 공정에서의 리드 타임을 추정하는 알고리듬이 필요하다. 비병목 공정 리드 타임의 정확도가 높아질수록 본 연구의 병렬 기계 알고리듬의 효과를 높여서 병목 공정의 효율을 더 높일 수 있다.

4. 참 고 문 헌

[1] Bertsekas, D. P., Tsitsklis, J. N., and Wu, C., "Rollout Algorithms for Combinatorial Optimization", *Journal of Heuristics*, vol. 3, pp. 245 - 262, 1997

[2] Hurink, J., and Knust, S., ""List Scheduling in a Parallel Machine Environment with Precedence Constraints and Setup Times", *Operations Research Letters*, Accepted 20 September, 2001

[3] Kim, J. I., and Lee, Y. H., "Batch Scheduling of Incompatible Job Families with Sequence Independent Setup Times", *Journal of the Korean Operations Research and Management Science Society*, vol. 26, pp. 69-83, 2001

[4] Lee, Y. H., and Pinedo, M., "Scheduling Jobs on Parallel Machines with Sequence-dependent Setup Times", *European Journal of Operational Research*, vol. 100, pp. 464-474, 1997

[5] Park, Y. S., Kim, S. Y., and Lee, Y. H., "Scheduling Jobs on Parallel Machines Applying Neural Network and Heuristic Rules", *Computers & Industrial Engineering*, vol. 38, pp. 189-202, 2000

[6] Rajaeskera, J. R., Murr, M. R., and So, K. C., "A Due-date Assignment Model for a Flow Shop with Application in a Lightguide Cable Shop", *Journal of Manufacturing System*, vol. 10, pp. 1-7, 1987

[7] Randhawa, S. U., and Smith, T. A., "An Experimental Investigation of Scheduling Non-Identical, Parallel Processors with Sequence-Dependent Set-up Times and Due dates", *International Journal of Production Research*, vol. 33, pp. 59 - 69, 1995

[8] Rothkopf, M. H., "Scheduling Independent Tasks on Parallel Processors", *Management Science*, vol. 12, pp. 437 - 447, 1966

저 자 소 개

백 종 관 : 현 서일대학 산업시스템경영과 교수.고려대학교 산업공학과 박사

관심분야는 생산 자동화 시스템, 경영정보시스템

김 형 준 : 현 서일대학 산업시스템경영과 교수로 재직 중

관심분야는 경제성 공학, 원가공학 등.