한국산학기술학회논문지
Vol. 10, No. 7, pp. 1702-1710, 2009

Rework 확률이 제품의 품질과 납기준수에 영향을 주는 공정을 위한 문제공간기반 탐색 알고리즘

강용하[1], 이영섭[2], 신현준[2*]
[1]노스캐롤라이나 주립대학교 산업공학과, [2]상명대학교 경영공학과

Problem space based search algorithm for manufacturing process with rework probabilities affecting product quality and tardiness

Yong Ha Kang[1], Young Sup Lee[2] and Hyun Joon Shin[2*]
[1]**Dept. of Industrial and Systems Engineering, North Carolina State University**
[2]**Dept. of management Engineering, Sangmyung University**

요 약 본 논문은 rework 발생확률을 고려하는 병렬기계 스케줄링 문제를 위해 문제공간기반 탐색 알고리즘을 제안한다. 각 기계와 작업유형별로 rework 발생확률이 존재하며 이것은 자동화된 공정에서 과거데이터로부터 산출가능하다. 스케줄링 문제의 데이터 벡터 (가공시간, 납기, 셋업시간, rework확률)를 교란시킴으로써 이웃해를 생성하고 이로부터 도출된 해는 EDDR이라는 효과적인 휴리스틱을 이용하여 평가한다. 제안된 알고리즘은 납기지연의 최대값과 rewok 발생 작업수로 평가함으로써 제품의 품질과 납기수준을 동시에 고려할 수 있도록 한다.

Abstract In this paper, we propose a problem space based search(PSBS) algorithm to solve parallel machine scheduling problem considering rework probabilities. For each pair of a machine and a job type, rework probability of each job on a machine can be known through historical data acquisition. Neighborhoods are generated by perturbing four problem data vectors (processing times, due dates, setup times, and rework probabilities) and evaluated through the efficient dispatching heuristic (EDDR). The proposed algorithm is measured by maximum lateness and the number of reworked jobs. We show that the PSBS algorithm is considerably improved from the result obtained by EDDR.

Key Words : Rework Probabilities, Parallel Machines, Scheduling, Problem Space Based Search

1. 서론

최근 제조업체들은 제품의 생산방식과 과정이 복잡해지고 다양해지면서 생산능력을 효율적으로 사용하는데 어려움을 겪고 있다[3]. 제조공정의 효율성을 방해하는 대표적인 요인들로는 불량발생으로 인한 재작업 (rework). 순서의존적인 작업준비시간(sequence - dependent setup time)과 작업투입시점 (release time) 등을 들 수 있다.

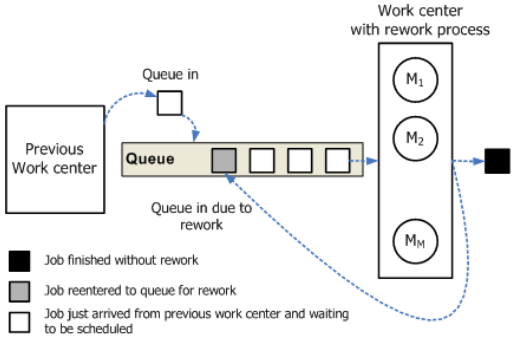
Rework는 반도체 산업뿐만 아니라 많은 제조 산업에서도 중요한 이슈이다[4]. 만일 공정에서 불량품이 발생한다면 재작업을 하거나 버려지게 되는데 이 때 발생하는 비용은 기업에게 부담을 가져올 수 있으므로 불량률을 줄이기 위한 효과적인 생산 관리가 필요하다. Uzsoy et al.[12,13]는 반도체 공정과 같은 고가의 설비를 운용하며, 순서의존적인 작업준비시간과 작업투입시점이 존재하는 제조공정은 일정계획을 수립할 때, 단순하고 직관적인 수작업 방식들을 사용한다면 생산성 및 비용측면에서 효과적인 결과를 기대하기가 힘들다고 밝히고 있다. 일반적으로 순서의존적인 작업준비시간이 존재하는 경우, 그 스케줄링 결과에 따라서 자원의 가동률과 그에 따른 납기준수율이 크게 좌우된다[5]. 작업투입시점은 일반적으

이 논문은 2007년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2007-313-D00901)
*교신저자 : 신현준 (hjshin@smu.ac.kr)
접수일 09년 05월 25일 수정일 09년 06월 25일 게재확정일 09년 07월 22일

로 BOM(bill of material)상의 하위부품조달 조달시점 또는 해당 작업의 전 공정 완료시점에 의해 결정되는데, 작업투입시점이 존재하는 제조환경에서는 기계의 유휴 시간을 최소화하기 위해 정확한 스케줄링이 필요하다.

Rework와 관련한 대부분의 논문들은 rework 제품의 재고관리와 재고정책에 따른 최적의 lot수량을 결정하는 것들이고 근래에 들어서 rework 정책을 dispatching규칙과 연관 지은 논문들이 발표되었다. Kuhl & Laubisch[6]는 반도체 공정에서 dispatching rule과 rework 전략이 생산성에 가장 큰 영향을 주는 두 가지 요소라고 간주하고 5가지의 dispatching 규칙과 세 가지 rework 전략 중 어떤 조합이 가장 높은 생산성을 얻을 수 있는 지에 관하여 시뮬레이션을 통해 비교 분석하였다. Sha et al.[9]는 반도체 포토공정(photolithography)의 dispatching을 통해 전체 공정이 운용된다고 보고 고객의 납기를 만족시키고 동시에 높은 제품품질을 얻기 위해서는 rework 전략과 dispatching 규칙을 적절하게 사용해야 한다고 말했다. 이 때 세 가지의 rework전략과 세 가지 dispatching 규칙의 조합으로 시뮬레이션을 수행하였다.

본 논문에서 논의할 문제를 정리해보면 다음과 같다. 병렬기계가 있고 다양한 종류의 제품 타입을 갖는 작업들이 기계에서 가공된다. 이때 기계와 제품 타입마다 가공 시 서로 다른 rework 확률이 존재하고, 제품타입별로 순서의존적인 작업준비시간과 작업별로 작업투입시점이 존재한다. Rework 확률은 작업이 어떤 기계에서 가공되는지에 따라서 달라질 수 있으며 이는 기계 자체의 특성에 의한 것이다. 만일 임의의 기계에서 가공을 마친 작업이 불량 판정을 받았을 경우, 양품 합격을 받을 때까지 rework 절차를 거친다. 즉 rework이 발생하지 않을 때가지 작업이 기계군 중 하나에서 반복적으로 투입, 가공과정을 거치게 된다. [그림 1]은 이와 같은 전체시스템을 설명하고 있다.



[그림 1] 전체 시스템 구성도

이상에서 논의된 문제를 효율적으로 해결하기 위해서는 1)rework 확률을 반영한 dispatching 알고리즘과 2)dispatching 알고리즘을 통해 생성된 작업투입순서를 보다 효과적으로 개선하기 위한 탐색알고리즘의 개발이 필요하다. 여기서 2)의 탐색을 위한 알고리즘은 자동화된 공정에서 요구되는 신속성을 감안해서 일정시간 내에 의사결정을 내릴 수 있도록 고안되어야 한다는 점이 중요하다. 따라서 본 논문에서는 rework 확률, 납기, 작업준비시간 등의 공정데이터의 속성을 이용하여 문제공간기반 탐색(problem space based search; 이하 PSBS) 기법을 사용하는 효율적인 알고리즘을 개발하는 것을 목표로 한다. 목적함수로는 납기준수와 품질비용 측면에서 납기지연시간(lateness time)의 최대값(이하 $L_{\max}$)을 최소화하는 것이고 다른 하나는 재가공 작업수(number of reworks; 이하 $NofR$)을 최소화하는 것이다. 본 논문에서 제안하는 PSBS 알고리즘의 성능을 평가하기 위해 초기해로 사용되는 rework을 고려한 dispatching 알고리즘과 비교하여 얼마만큼의 성능 향상을 보여주는지 실험을 통해 보여준다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 본 논문이 다루고 있는 문제를 정의하였고 3장에서는 본 논문이 제안한 PSBS 병렬기계 알고리즘에 대하여 기술되어 있고 4장에서는 다양한 환경 하에서 실행된 실험 결과 및 분석을, 마지막으로 5장에서 결론 및 추후 연구 과제를 제시한다.

2. 문제 정의

본 연구에서는 $L_{\max}$와 $NofR$을 각각 최소화하기 위해 rework 가능성이 존재하는 상황에서 N개의 작업을 $M(m=1,\cdots,M)$개의 병렬기계에 스케쥴링하는 알고리즘을 제시한다. N개의 작업은 같은 제품타입(product type)을 갖는 C개의 작업군으로 나뉘고, 이 때 작업군 J_c를 제품타입이 c인 작업들의 집합이라 한다. 작업군 J_c에 속한 작업들의 인덱스를$j(j=1,\cdots,N_c)$라고 했을 때, 이들 작업 j는 각각 납기(due date) d_j와 작업투입시점(release time) r_j를 갖는다. 각 작업 j의 가공시간(processing time) p_j는 작업들이 기계에서 처리되는 순서 및 가공 기계와는 독립적으로 주어지지만 작업준비시간은 작업들이 가공되는 순서와 속한 작업군에 따라 달라지는 순서의존적인 준비시간(sequence dependent setup times)을 갖는다. 여기서 순서의존적인 작업준비시간이란 작업군 J_c에 속

한 작업 i의 가공을 마친 후 작업군 J_c 에 속한 작업 j를 준비하는 데 소요되는 시간을 $s_{c'c}$라 할 때, $s_{c'c} \neq s_{cc'}$ 인 성질을 갖는 작업준비시간을 의미한다. 물론 작업 i와 j가 같은 제품군에 속한다면 $s_{c'c}=0$이다. 이 때 작업군 J_c에 속한 작업 j가 기계 m에서 가공을 한 후 불량 판정을 받아서 재가공을 할 확률은 P_{cm}이다.

t라는 시점에 기계 m이 가공 중이던 작업 $i(i \in J_c)$의 가공을 완료한 후 유휴(idle)상태가 되어 투입작업으로 작업 $j(j \in J_c)$가 선택되었다고 하자. 현재 rework확률이 존재하기 때문에 작업 j는 기계 m에서 가공을 완료한 후 다시 가공을 해야만 하는 상황이 발생할 수도 있다. 만일 작업 j가 기계 m에서 가공을 완료한 후 rework판정을 받지 않을 경우에는 작업준비시간과 가공시간의 합인 $s_{c'c}+p_j$만큼의 시간이 소요되지만 rework 판정을 받을 경우에는 $s_{c'c}+p_j$뿐만 아니라 재가공에 필요한 시간이 더 필요하게 된다. 이 시간을 R_j라고 한다면 R_j은 재작업 판정을 받은 작업이 기계 앞으로 돌아가 대기하다가 다시 기계에 투입될 때까지의 추정 체재시간(Estimated Sojourn Time)이라고 할 수 있다. Rework가 발생하지 않아서 가공을 바로 마칠 수 있을 확률은 (1- P_{cm})이고 rework가 발생하여 다시 기계 앞으로 돌아가 대기할 확률은 P_{cm}이므로 결국 평균적으로 작업 j를 기계 m에서 가공할 경우 가공을 위해 필요한 시간은 (1- P_{cm}) $(s_{c'c}+p_j)+P_{cm}(s_{c'c}+p_j+R_j)$라고 할 수 있다. 이를 예상가공시간(Expected Processing Time, 이하 *EPT*)이라고 하자. 기계 m에 작업 j를 투입하는 시점이 t이므로 작업 j는 t+(1- P_{cm})$(s_{c'c}+p_j)+P_{cm}(s_{c'c}+p_j+R_j)$에 가공을 완료 하게 되는데 이를 예상완료시간(Expected Completion Time, 이하 *ECT*)이라고 한다.

$$EPT=(1- P_{cm})(s_{c'c}+p_j)+P_{cm}(s_{c'c}+p_j+R_j)$$
$$=(s_{c'c}+p_j)+P_{cm}R_j \quad (1)$$

$$ECT=t+(1-P_{cm})(s_{c'c}+p_j)+P_{cm}(s_{c'c}+p_j+R_j)$$
$$=t+(s_{c'c}+p_j)+P_{cm}R_j \quad (2)$$

[Notation]

$j(j=1 \sim N)$: 작업 인덱스

$c(c=1 \sim C)$: 제품 타입 인덱스

J_c : 제품타입이 c인 작업들의 작업군 인덱스

$m(m=1 \sim M)$: 병렬기계 인덱스

P_{cm} : 작업군이 J_c인 작업 j를 기계m에서 가공했을 때 재가공확률(rework probability)

p_j : 작업 j 의 가공 시간(processing time)

$s_{c'c}$: 제품타입이 c인 작업 j가 투입되기 직전에 그 기계에서 가공을 완료한 작업의 제품타입이 c' 일 때 작업 준비시간(setup time)

d_j : 작업 j의 납기(due date)

r_j : 작업 j의 투입 가능 시간(release time)

R_j : rework판정을 받은 작업 j의 추정체재시간

$\overline{s_{.c}}$: 제품타입이 c인 작업 j가 투입되기 직전에 그 기계에서 가공을 완료한 임의의작업의 제품타입에 대한 평균작업 준비시간(average setup time), 즉 $\overline{s_{.c}}=\frac{1}{|C|}\sum_{\forall c} S_{c'c}$

재가공시 체재 시간인 R_j는 rework시 한 번의 가공이 더 발생하게 되므로 R_j는 작업을 가공 시 필요한 시간인 $\overline{s_{.c}}+p_j$보다 같거나 큰 시간이라고 할 수 있다. 왜냐하면 작업 j보다 우선순위가 높은 다른 작업들이 있다면 바로 기계에 투입되지 못하여 대기하는 시간이 발생할 수도 있기 때문이다. 그러므로 재가공하는데 소요되는 시간인 R_j는 $NR\times(\overline{s_{.c}}+p_j)$로 나타낼 수 있다. 여기서 NR은 재가공시간 결정모수로서 본 연구에서는 실험을 통하여 NR값을 결정하여 사용한다.

3. 제안 알고리즘

문제공간기반탐색(Problem space based search : PSBS)은 Storer et al.[10]에 의해 처음 제안된 이웃해 생성 방법으로 많은 스케쥴링 문제에서 그 성과를 보여주고 있다. Leon & Ramamoorthy는 자원제약 스케쥴링 문제(resource constrained scheduling)[7]와 유연흐름생산라인 스케쥴링 문제(flexible flow line scheduling)[8]에서 PSBS는 우수한 성능을 보일 뿐 아니라 여러 가지 스케쥴링 문제에서 다양하게 적용될 수 있다는 점을 보여주었다. Avic et al.[2]는 단일 기계에서 전체 가중 납기지연값을 최소화하는 문제를 PSBS를 통해 해결하였고 Turkcan & Akturket[11]은 생산비용과 전체 가중 납기 지연값을 통시에 최소화하는 문제에서 PSBS를 적용하여 효과적인 해를 제시하였다.

본 논문에서는 병렬기계에서 재작업 확률(rework probability), 순서의존적인 작업준비시간과 작업투입 시점이 존재할 때 PSBS를 이용하여 $L_{\max}$와 $NofR$를 최소

화하기 위한 효과적인 스케쥴링 방안을 제시하고자 한다.

3.1 문제공간기반 이웃해

이 장에서는 Storer et al.[10]가 제안한 이웃해 생성 방법인 문제공간기반 이웃해(Problem space based neighborhoods: 이하 PSBN)에 관하여 기술한다.

다양한 국지탐색기법(local search method)이 근사알고리즘에 적용되어 왔는데 대표적인 국지탐색기법으로는 타부탐색(tabu search), 유전알고리즘(genetic algorithm), 시뮬레이티드 어닐링(simulated annealing) 등이 있다[1].

스케쥴링 문제에서 해를 구한다는 것은 일반적으로 기계에 투입할 작업들의 순서를 정해주는 것이라고 할 수 있다. 기존 국지탐색방법들이 스케쥴링 문제에서 적용될 때 정해진 해(작업순서)안에서 작업들의 위치를 서로 바꾸거나(swap) 어떤 작업을 다른 위치로 이동(insert)시킴으로써 즉, 기존의 작업 순서(job sequence)를 변경하여 새로운 스케쥴을 생성하는 방식으로 이웃해를 생성시킨다. 이것을 해 공간 기반 이웃해(solution space based neighborhoods)라고 할 수 있다. 대부분의 국지탐색방법들이 이런 식으로 최적해 값을 찾거나 일정 시간이 경과할 때까지 탐색을 계속 진행해 나간다.

반면 Storer et al.[10]가 제안한 PSBN의 방식은 정해진 해를 기반으로 이웃해를 생성하는 것이 아니라 문제가 가지고 있는 특성을 이용하여 이웃해를 생성한다. 이것이 바로 PSBN이다. PSBN은 문제가 가지고 있는 특성요소를 변동시킨 후 문제에 특화된 휴리스틱을 이용하여 작업 순서를 만들어줌으로써 이웃해를 생성해 나간다. 이렇게 생성된 이웃해를 평가할 때는 변형되기 전 문제특성요소를 새로 만들어진 작업 순서에 적용하여 구해진 값을 이용한다.

스케쥴링 문제에서 f를 문제를 구성하는 요소의 데이터 값(예 : 가공시간, 납기 등)의 벡터라고 하고 h를 문제에 맞는 휴리스틱이라고 하자. 이때 생성되는 해(작업순서) π는 $h(f)$라고 정의할 수 있다.

PSBN에서는 데이터 값 벡터인 f를 변형(perturbation)시키고 이를 휴리스틱에 적용하여 새로운 이웃해(작업순서)를 만든다. 이 때 변형된 데이터 벡터들의 집합 F는 식 (3)과 같다.

$$F=\{f_i=f_o+u_i,\ i=1,...,m\} \tag{3}$$

이때, f_o는 문제의 원래 데이터 값의 벡터이고 u_i는 임의로 생성된 데이터 값을 변동시키는 벡터로 보통 균일분포(uniform distribution) $U(-\theta f_o\ ,\ \theta f_o)$로부터 생성된다. θ는 변동폭을 조절하는 모수(perturbed range control parameter)로서 그 값은 PSBS 성능에 큰 영향을 끼치게 되므로 적절한 값을 찾아서 이용하여야 한다[12]. 만일 θ가 0이면 perturb 성질은 사라지게 되어 원래의 데이터를 base heuristic을 이용하여 푼 것이 된다. 반면에 θ가 무한대로 커지면 데이터가 심하게 변동이 되기 때문에 임의로 작업 순서를 결정하게 되는 효과를 주게 된다.

이렇게 만들어진 m개의 데이터 벡터 f_i를 휴리스틱인 h에 적용하면 m개의 새로운 이웃해(작업순서)가 얻어지게 된다. 즉, 이웃해 집합 Π는 다음의 식 (4)와 같다.

$$\Pi=\{\pi_i=h(f_i),\ i=1,...,m\} \tag{4}$$

변동 데이터 값은 단지 새로운 작업순서를 얻기 위한 것이고 목적함수 값을 평가할 때는 적절하지 않다. 실제 필요한 목적함수 값을 구하기 위해서는 변동되기 전의 데이터가 필요하다.

v를 목적함수 값이라고 한다면 $v=\pi(f)$로 정의할 수 있으며 목적함수 값의 집합 V는 다음의 식 (5)와 같다.

$$V=\{v_i=\pi_i(f_o),\ i=1,...,m\} \tag{5}$$

이렇게 구해진 v_i중에서 문제가 요구하는 목적함수에 따라 $\max(\text{or }\min)v_i$가 목적함수 값으로 최종적으로 선택된다.

3.3 베이스 휴리스틱 (h)

Leon & Ramamoorthy[9,10]에 따르면 PSBS에 사용되는 heuristic은 다음의 두 가지 요구사항을 충족시켜야 한다. 첫 번째로 좋은 해 값을 보장해야 하며 두 번째로 계산시간이 짧아야 한다. PSBS의 매 반복마다 휴리스틱 h가 사용되기 때문에 만일 사용되는 h의 성능이 좋지 않다면 이는 임의 탐색(random search)과 다를 바 없게 된다. 또한 계산 시간도 매 탐색마다 h에 걸리는 시간이 포함되기 때문에 만일 h 자체의 탐색 시간이 길어진다면 전체 탐색 시간은 증가하게 될 것이다.

따라서 본 논문에서는 위의 사항들을 고려하여 dispatching 알고리즘인 EDDR (Earliest Due Date with Rework probability)을 휴리스틱 h로 제안한다. 본 논문에서 제안하는 PSBS 알고리즘에 대한 자세한 설명에 앞서 EDDR 휴리스틱을 간략히 설명하면 다음과 같다.

EDDR 알고리즘은 유휴(idle)상태인 기계가 발생할 경우 적용된다. 이때 그 기계에서 가공했을 때 rework 확률이 가장 작은 작업타입인 작업들이 속한 작업군을 선호

작업군이라 하고 그렇지 않은 작업군을 비선호작업군이라고 한다. 마찬가지로 어떤 작업타입을 가공했을 때 rework될 가능성이 적은 기계를 선호기계라 하며 그렇지 않은 기계를 비선호기계라 한다.

[EDDR 알고리즘 절차]

Step 1. 작업분류

1. 현재 queue에 있는 작업들을 작업타입별로 분류
   기계가 유휴상태가 된 시점을 t라 할 때 $r_j \le t$인 작업 j와 rework판정을 받아서 기계 앞으로 되돌아와서 대기 중인 작업 j를 작업타입별로 분류한다.
2. 작업타입 별로 분류된 작업들을 납기가 빠 른 순서(Earliest Due Date:EDD)로 정렬

Step 2. 대상 작업군 Ω 선정

1. 선호작업군
   step 1에서 EDD로 정렬된 작업들 중 가장 선두에 있는 작업을 Ω에 포함
2. 비선호작업군
   모든 비선호작업군에 대하여 선호기계에서 먼저 투입된 작업의 가공이 종료되는 순간까지 기다린 후, (a)가공을 했을 때의 예상 완료시간(ECT)과 (b)현재 유휴(idle)상태가 된 기계에서 가공했을 때의 예상 완료시간(ECT)을 비교하여 (a)>(b)인 작업들 중 가장 EDD인 작업을 선택하여 Ω에 포함

Step 3. 대상 작업군에서 투입할 job 선택

1. Ω에 포함된 작업들의 현재 유휴 상태인 기계에서의 예상 완료 시간(ECT)을 계산
2. 위 단계에서 계산한 예상 완료 시간이 가장 빠른 작업을 기계에 투입

END.

Step 1은 현재 스케쥴링 시점 t에서 기계 앞에서 자신의 가공 순서를 기다리는 작업들(새로 주문된 작업들과 rework 판정을 받고 다시 기계 앞으로 돌아온 작업들)을 EDD로 정렬시킴으로써 현재 타입별로 가장 급한 작업이 무엇인지 알기위한 단계이다.

Step 2는 step 1에서 분류된 작업 순서에 의해 선호작업군과 비선호작업군에서 각각 후보 작업들을 선별하는 단계이다. step 3에서는 step 2에서 선별된 후보 작업들을 대상으로 실제 기계에 투입될 작업을 선정하는 단계이다. 대상작업군 Ω에 포함된 작업들을 현재 유휴한 기계에서 가공했을 때 각 작업들의 ECT를 계산하여 그 중 가장 작은 ECT를 갖는 작업을 기계에 투입할 작업으로 최종 선정하고 유휴 기계에 투입한다. 즉, rework확률을 고려하여 대상 작업의 완료 시간을 계산하여 가장 빨리 작업을 완료할 수 있는 작업을 기계에 투입하는 것이다.

3.4 문제공간기반탐색 알고리즘

실험에 의하면 단순히 EDDR을 이용하여 구한 해는 다른 할당 규칙인 EDD, MS, ATCS 보다는 좋은 해를 보여주지만, dispatching 알고리즘이라는 근시안적 특성의 한계를 벗어나지 못하기 때문에 최적해에 근접한 해(Near optimal solution)를 보장해 주지 못한다. 따라서 본 연구에서는 문제공간기반탐색 알고리즘(problem Space Based Search : PSBS)을 통해 EDDR의 한계점을 보완하여 효과적인 생산계획을 수립하고자 한다.

앞의 두 절에서는 문제공간(PS)의 개념과 PSBS에서 사용될 기본 휴리스틱 방법에 대하여 알아보았다. 본 절에서는 PSBS 알고리즘에 관하여 설명하고자 한다. PSBS는 목적함수가 $L_{\max}$와 $NofR$을 최소화시키는 것이기 때문에 탐색방법으로 최대 하강법(steepest descent method)을 이용한다. 즉, 어떤 데이터를 기준으로 일정 횟수의 이웃해를 생성한 후 그 중 최고의 목적함수 값을 갖는 이웃해를 찾아내고 다시 그 이웃해의 데이터를 기준으로 탐색을 반복한다. 이런 방법으로 정해진 탐색 횟수만큼 탐색을 완료하면 알고리즘은 종료된다.

NOS를 변동을 시킬 때 기준이 되는 데이터의 개수라 하고, NOI를 그 데이터를 기준으로 생성되는 이웃해의 개수라고 하면 총 탐색 횟수는 $NOS \times NOI$가 된다.

다음은 위에서 설명한 문제공간기반 탐색알고리즘의 탐색방법이다.

[PSBS algorithm 절차]

Step 0. (초기화)

기준 데이터, 목적함수 값 설정

$f^S \leftarrow f_o,\ v_B \leftarrow \pi_o(f_o)$

Do (S=1 to NOS)

Do (i= 1 to NOI)

Step 1. (데이터 변동)

기준 데이터로 변동데이터 생성

$f_i \leftarrow f^S + u_i$

Step 2. (이웃해 생성)

변동데이터를 EDDR에 적용하여 이웃해 생성

$\pi_i = h(f_i)$

Step 3.(목적함수 값 평가)

생성된 이웃해에 원래 데이터를 대입하여 목적함수 값 계산

$v_i = \pi_i(f_o)$

계산된 목적함수가 현재 최우수 목적함수 값보다 작으면 최우수 목적함수 값 변경하고 그때의 변동 데이터를 최우수 데이터로 설정,

$v_i < v_B$이면 $v_B \leftarrow v_i$, $f^B \leftarrow f_i$

End.

Step 4. (기준 데이터 값 변경)

S를 1증가시키고 기준데이터 값 변경

$S \leftarrow S+1$, $f^S \leftarrow f^B$

End.

Step 0은 데이터를 변동시키기 위하여 기준이 데이터를 설정하는 단계이다. 여기서 원래의 문제 데이터 벡터인 f_o가 탐색의 기준 데이터 벡터 $f^S(S=1 \sim NOS)$가 된다. 목적함수의 경우 PSBS를 적용하기 전 EDDR을 사용하여 구한 값을 초기 목적함 수 값으로 놓는다. 즉, 원래의 문제 데이터 벡터 f_o를 EDDR에 적용하여 생성한 이웃해 π_o에 f_o를 대입한 값인 $\pi_o(f_o)$가 초기 목적함수 값이 된다.

다음으로 Step 1부터 Step 3까지가 생성되는 이웃해의 수인 NOI만큼 반복된다. Step 1은 기준 데이터를 변동시키는 과정이다. 3.1 절에서 설명한 것처럼 기준데이터 f^S에 변동값 u_i를 더하여 변동 데이터 f_i를 생성한다. Step 2에서는 f_i를 기본 휴리스틱인 EDDR에 적용하여 새로운 이웃해 π_i를 생성한다. 목적함수 값을 평가할 때는 원래 데이터인 f_o를 이용해야 하므로 Step 3에서는 π_i에 f_o를 대입하여 목적함수 값 v_i를 구한다. 만일 그 값이 현재 최우수 목적함수 값보다 더 우수하다면(작다면) 최우수 목적함수 값 v_B는 v_i로 바뀌게 되고 그때의 f_i를 최우수 변동 데이터 값인 f^B로 설정한다.

Step 1부터 Step 3까지의 탐색을 완료하면 기준 데이터 값을 변경하고 위의 탐색 과정을 동일한 방식으로 반복해 나간다. Step 4에서는 기준 데이터 값 f^S를 최우수 목적함수 값을 구할 때의 이웃해를 생성한 데이터 값 f^B로 치환한다. 초기에 설정한 NOS만큼의 기준 데이터에 대하여 탐색이 끝나면 알고리즘은 종료된다.

본 논문에서는 다루고 있는 문제에서 스케쥴링 결과에 영향을 끼치는 데이터 벡터들은 납기, 가공시간, rework 확률 그리고 순서의존적인 작업준비시간의 4개의 벡터로 추정할 수 있다. 4장에서는 각 데이터 벡터별로 PSBS를 수행한다. 그리고 휴리스틱 h로 사용된 EDDR과 비교하여 얼마만큼의 성능 향상을 보이는지를 평가하도록 한다.

4. 결과 분석

4.1 실험 설계

본 연구에서 제시한 알고리즘의 효과를 검증하기 위하여 동일한 실험 환경 상태에서 모의실험을 실시하였다. 앞에서도 언급했듯이 4가지 문제 구성 요소인 납기(due date; D), 가공시간(processing time; P), 재가공확률(rework probability; RP) 그리고 순서의존적인 작업준비시간(sequence dependent setup time; S)에 대하여 PSBS를 시행하였고 실험결과들을 이용하여 알고리즘의 성능과 가장 유의한 문제요소가 무엇인지에 관하여 분석하였다. 알고리즘의 성능을 비교하기 위한 척도로는 $L_{\max}$와 $NofR$이 사용된다.

실험에 사용된 데이터의 생성 기준은 다음과 같다. 가공 시간과 작업 준비 시간은 U[150,200]의 범위를 갖는 균일분포로부터 생성하였고, 작업투입시점은 0으로부터 평균최대완료기간(Expected makespan : T)의 R배까지인 [0,R*T]의 범위를 갖는 균일분포로부터 생성하였다. 여기서 T는 평균작업준비시간과 평균가공시간의 합을 작업수 N과 곱한 후 기계수 M으로 나누어준 값이며, R은 작업투입시점의 범위 모수(Release Time Range Parameter)로서 본 실험에서는 고정값으로 1.0이 사용되었다. 납기는 식(6)에 의해 생성되며, 수식에 사용된 α값은 [-1,4]의 범위를 갖는 균일분포로부터 발생된다.

$$d_i = r_i + 2\alpha p_i \quad (6)$$

본 연구의 핵심 요소인 rework확률은 [표 2]에 정리된 것처럼 작업들의 기계선호도에 따라 확률값을 B(Best), N.B(Not bad), P(Poor)의 세 가지 단계로 나누었고 각각 [0,0.001], [0.1, 0.2], [0.2, 0.3]의 범위를 갖는 균일분포로부터 생성하였다.

본 실험에서 사용하는 벤치마킹데이터는 [표 1]과 같이 작업수와 제품타입, 그리고 변동될 문제요소의 조합으로 생성된 320개의 문제로 구성되어 있다. 단, 기계수는 3대로 고정시켰다.

[표 1] 실험계획

	Values	Total
No. of Jobs	100, 500, 1000, 2000	4
No. of Job Types	5,10	2
No. of perturbed Factors	D, P, RP, S	4
Combination		32
Problems per Combination		10
Total Problems		320

PSBS에서 사용되는 모수로는 변동폭 조절 모수 θ, 기준데이터 수 *NOS*와 탐색횟수 *NOI*가 있는데 초기 실험을 통해 그 값을 각각 0.25, 5, 100으로 고정하여 사용하였다. 본 연구에서 제시한 알고리즘은 모두 C#를 이용하여 구현하였고, 펜티엄4 2.4 GHz 컴퓨터에서 실험하였다.

4.2 실험 결과 및 분석

이 장에서는 4.1에서 언급한 320개의 실험 데이터를 사용하여 본 논문이 제시한 PSBS 알고리즘의 성능을 두 가지 목적함수인 납기지연시간(lateness time)의 최대값($L_{\max}$)과 재가공된 작업의 수(Number of reworked jobs : *NofR*)의 최소화에 대해서 분석해 보기로 한다.

4.2.1 PSBS algorithm 성능 분석

■ 목적함수: $L_{\max}$

[표 3]는 base heuristic인 EDDR과 4가지 문제 요소에 PSBS를 적용한 결과를 보여주고 있다. 표에서 보듯이 모든 문제에서 4가지 요인을 변동시켜서 PSBS를 적용한 것이 EDDR보다 좋은 해값을 주는 것으로 나타난다. 문제별로 약간의 차이가 있지만 대체적으로 순서의존적인 작업준비시간과 재작업확률을 변동시킨 경우가 나머지 경우보다 좋은 해값을 보여준다.

[표 2] rework 확률

Type \ MC	1	2	3	4	5	6	7
A	B	P	N.B	N.B	N.B	N.B	N.B
B	N.B	B	P	N.B	N.B	N.B	N.B
C	P	N.B	B	N.B	N.B	N.B	N.B
D	N.B	N.B	N.B	B	P	N.B	N.B
E	N.B	N.B	N.B	P	B	N.B	N.B
F	N.B	N.B	N.B	N.B	N.B	B	P
G	N.B	N.B	N.B	N.B	N.B	P	B
H	B	N.B	N.B	N.B	P	N.B	N.B
I	P	N.B	N.B	N.B	B	N.B	N.B
J	N.B	B	N.B	N.B	N.B	P	N.B

[표 3] $L_{\max}$에 대한 실험결과 값

작업수	제품타입수	EDDR	D	P	R P	S
100	5	4,815	3,115	2,706	3,003	2,533
		(995)	(472)	(215)	(419)	(181)
	10	6,220	4,551	4,633	5,423	3,917
		(492)	(465)	(238)	(627)	(468)
500	5	14,295	12,754	10,309	9,069	8,197
		(3,684)	(615)	(645)	(604)	(426)
	10	32,041	22,633	22,026	22,842	14,501
		(2,966)	(1,606)	(855)	(1,569)	(617)
1000	5	26,366	22,386	16,257	13,136	12,893
		(6,576)	(1,644)	(1,427)	(754)	(553)
	10	57,198	39,500	44,539	37,516	24,120
		(5,566)	(1,994)	(1,956)	(2,112)	(1,645)
2000	5	43,902	40,443	32,330	25,758	22,829
		(9,947)	(5,498)	(2,413)	(1,864)	(1,646)
	10	114,335	79,799	79,973	70,733	41,599
		(15,954)	(4,426)	(2,126)	(6,041)	(2,415)

[표 4]는 PSBS 알고리즘을 사용했을 때 소요되는 평균 CPU 시간(Final CPU time)과 최고해까지 도달하는 데 걸리는 최고 CPU 시간(Best CPU time)을 나타내고 있다. 시간도 마찬가지로 순서의존적인 작업준비시간과 재작업 확률을 변동시켰을 경우가 나머지 2가지 경우보다 빠른 시간 안에 전체 수행을 마쳤다. 이는 순서의존적인 작업준비시간이나 재작업 확률은 작업수 만큼의 데이터가 변동되는 것이 아니라 제품 타입수 만큼이 변동이 되기 때문에 상대적으로 변동되는 양이 작아져서 시간이 감소되는 것으로 추정된다.

■ 목적함수: *NofR*

[표 5, 6]은 각각 목적함수가 *NofR*인 경우의 목적함수 값과 CPU 시간에 관한 결과를 보여주고 있다. 표에서 보듯이 목적함수가 *NofR*일 때도 모든 문제에서 $L_{\max}$일 때와 비슷한 결과를 보여준다. 문제별로 약간의 차이가 있지만 대체적으로 순서의존적인 작업준비시간과 재작업확률을 변동시킨 경우가 나머지 경우보다 좋은 해값을 보여준다.

5. 결론 및 추후연구

본 연구에서는 작업이 속한 제품타입과 가공되는 기계에 따라 rework확률이 달라지는 병렬기계 문제에서 $L_{\max}$와 *NofR*의 두 가지 목적함수를 최소화하는 PSBS 알고리즘을 제시하였다. 두 목적함수에 대하여 4가지 요인별

로 각각 실험을 시행하였고, 그 결과 PSBS 알고리즘이 동일한 문제에 대해 최근 연구에서 효율적 dispatching 알고리즘으로 제시된 EDDR보다 매우 우수한 해를 산출한다는 것을 보였다. 또한 문제 요인별로는 대부분의 경우에 순서의존적인 작업준비시간과 재작업 확률을 변동시켰을 경우가 좋은 해를 산출하였으며 가장 유의한 요소로 밝혀졌다.

추후연구로는 job shop과 같은 매우 복잡한 작업장에서 실시간적으로 효율적인 dispatching 규칙을 선정해서 사용해야 될 때, 본 연구에서 제시한 PSBS 알고리즘의 문제 데이터 벡터 분석 방법을 활용하는 방안을 적용하는 것이 있다. 또한 다른 메타휴리스틱 탐색방법들과의 비교실험을 통하여 PSBS의 성능을 세밀하게 분석하여 성능 개선 가능 여부를 파악할 필요가 있다.

[표 4] $L_{\max}$에 대한 최고해 도달 시간과 평균시간

Problem	Best CPU time				Final CPU time			
	D	P	R P	S	D	P	R P	S
100_5	2.6	3.0	1.1	2.9	5.2	5.4	5.2	5.4
	(1.4)	(1.7)	(1.4)	(1.5)	(0.5)	(0.3)	(0.5)	(0.4)
100_10	3.8	3.7	1.6	3.9	6.6	6.9	6.7	6.9
	(2.1)	(1.9)	(1.6)	(1.9)	(0.6)	(0.4)	(0.6)	(0.5)
500_5	14.7	12.1	40.8	31.1	63.8	78.3	59.5	68.6
	(16.9)	(17.3)	(13.6)	(18.8)	(3.3)	(4.3)	(4.1)	(4.5)
500_10	82.9	61.9	57.7	87.8	135.9	143.0	137.8	124.2
	(33.4)	(42.1)	(48.3)	(24.2)	(5.7)	(4.6)	(4.5)	(4.4)
1000_5	60.2	80.1	165.5	148.2	265.8	305.5	244.6	254.9
	(69.8)	(45.5)	(38.6)	(58.4)	(10.5)	(12.5)	(11.7)	(12.4)
1000_10	218.9	242.7	301.3	288.8	507.4	591.2	475.2	433.2
	(207.2)	(191.6)	(119.5)	(115.1)	(14.5)	(14.3)	(12.6)	(15.4)
2000_5	62.4	187.3	433.0	727.5	1059.1	1299.2	965.1	1020.8
	(85.4)	(160.1)	(295.6)	(193.7)	(29.1)	(112.3)	(31.0)	(69.9)
2000_10	1070.0	1117.3	1025.8	1213.9	2219.3	2766.0	2163.1	1704.3
	(741.8)	(685.3)	(623.3)	(483.7)	(59.7)	(322.7)	(84.7)	(63.0)

[표 5] *NofR*에 대한 실험결과 값

작업수	제품타입수	EDDR	D	P	R P	S
100	5	16	8	7	9	6
		(3)	(3)	(3)	(3)	(3)
	10	18	9	7	12	8
		(4)	(4)	(3)	(4)	(3)
500	5	59	42	44	34	39
		(6)	(4)	(4)	(4)	(3)
	10	82	60	59	54	50
		(8)	(4)	(3)	(7)	(6)
1000	5	115	80	76	60	67
		(14)	(7)	(7)	(9)	(7)
	10	159	127	123	106	103
		(17)	(7)	(7)	(10)	(7)
2000	5	211	158	158	107	132
		(22)	(11)	(11)	(12)	(10)
	10	319	265	263	227	208
		(22)	(13)	(13)	(21)	(11)

[표 6] *NofR*에 대한 최고해 도달 시간과 평균시간

작업수	제품타입수	Best CPU time				Final CPU time			
		D	P	R P	S	D	P	R P	S
100	5	1.2	1.2	0.8	2.6	5.2	5.4	5.2	5.3
		(1.1)	(1.4)	(1.0)	(1.8)	(0.5)	(0.3)	(0.6)	(0.3)
	10	3.3	3.1	1.5	3.1	6.9	7.0	6.6	6.8
		(2.3)	(2.4)	(1.6)	(2.0)	(0.6)	(0.5)	(0.6)	(0.5)
500	5	15.9	22.9	35.0	41.3	63.4	78.8	57.2	71.1
		(10.1)	(24.9)	(17.4)	(26.0)	(3.5)	(4.7)	(3.6)	(3.0)
	10	83.4	72.6	103.9	91.6	134.7	143.8	135.7	121.7
		(33.2)	(47.0)	(36.1)	(28.4)	(6.6)	(4.8)	(5.9)	(5.6)
1000	5	123.2	188.5	197.5	190.4	261.3	321.7	238.5	255.7
		(93.6)	(90.2)	(37.3)	(42.5)	(10.3)	(20.7)	(9.8)	(13.5)
	10	142.3	323.9	307.3	232.3	510.1	601.1	468.2	444.3
		(122.5)	(215.2)	(142.0)	(145.0)	(14.7)	(31.3)	(16.3)	(30.5)
2000	5	649.5	539.1	714.2	729.8	1121.0	1253.7	915.5	964.0
		(397.1)	(413.4)	(162.9)	(148.1)	(52.4)	(31.0)	(48.4)	(42.7)
	10	875.4	1414.6	1489.8	1080.9	2203.1	2740.9	2140.2	1780.8
		(638.0)	(770.5)	(727.0)	(343.9)	(63.1)	(150.0)	(104.7)	(93.9)

참고문헌

[1] 김여근, 윤복식, 이상복, [메타휴리스틱], 영진출판사, 1997.

[2] Avic, S., M.S. Akturk and R.H. Storer, "A problem space algorithm for single machine weighted tardiness problems", IIE Transactions, Vol.35, pp.479-486, 2003.

[3] Demirkov, E. and R. Uzsoy, "Decomposition methods for reentrant flow shops with sequence-dependent setup times", Journal of Scheduling, Vol.3, pp.155-177, 2000.

[4] Flapper, S.D.P., J.C. Fransoo, R.A.C.M. Broekmeulen and K. Inderfurth, "Planning and control of rework in the process industries : review", Production Planning & Control, Vol.13, No.1, pp.26-34, 2002.

[5] Kim, S.C. and P.M. Bobrowski, "Impact on sequence-dependent setup time on job shop scheduling performance", International Journal of Production Research, Vol.32, No.7, pp.1503-1520, 1994.

[6] Kuhl, M.E. and G.R. Laubisch, "A simulation study of dispatching rules and rework strategies in semiconductor manufacturing", IEEE/SEMI Advanced Semiconductor Manufacturing Conference, 2004.

[7] Leon,V.J. and B. Ramamoorthy, "strength and adaptability of problem space based neighborhoods

for resource-constrained scheduling", Operations Research Spektrum, Vol.17, No.2, pp.173-182, 1995.
[8] Leon, V.J. and B. Ramamoorthy, "An adaptable problem-space based search method for flexible flow line scheduling", IIE Transactions, Vol.29, pp.115-125, 1997.
[9] Sha, D.Y., S.Y. Hsu, Z.H. Che and C.H. Chen, "A dispatching rule for photolithography scheduling with an online rework strategy.", Computers & Industrial Engineering, Vol.50, pp.233-247, 2006.
[10] Storer, R.H., S.D. Wu and R. Vaccari, "New search spaces for sequencing problems with application to job shop scheduling", Management Science, Vol.38, No.10, pp.1495-1509, 1992.
[11] Turkcan, A. and M.S. Akturk, "A problem space algorithm in multi-objective optimization", Journal of Intelligent Manufacturing, Vol.14, pp.363-378.
[12] Uzsoy, R., C.Y. Lee and L.A. Martin-Vega, "A review of product planning and scheduling models in the semiconductor industry Part I : Systems characteristics, performance evaluation and production planning", IIE Transaction on Scheduling and Logistics, Vol.24, No.4, pp.47-61, 1992.
[13] Uzsoy, R., C.Y. Lee and L.A. Martin-Vega, "A review of product planning and scheduling models in the semiconductor industry Part II : Shop-floor control", IIE Transaction on Scheduling and Logistics, Vol.26, No.5, pp.44-55, 1994.

강 용 하(Yong Ha Kang) **[정회원]**

- 2001년 2월 : 고려대학교 산업시스템정보공학과(공학사)
- 2003년 2월 : 고려대학교 산업시스템정보공학과(공학석사)
- 2008년 2월 : 고려대학교 산업시스템정보공학과(공학박사)
- 2008년 5월 ~ 현재 : 미국 노스캐롤라이나 주립대학교 산업공학과 Post-Doc

<관심분야>
생산관리, 공급사슬망관리, 스케줄링

이 영 섭(Young Sup Lee) **[정회원]**

- 1987년 2월 : 고려대학교 산업공학과(공학사)
- 1989년 2월 : 한국과학기술원 산업공학과(공학석사)
- 2007년 3월 ~ 현재 : 상명대학교 경영공학과 박사과정

<관심분야>
생산관리, 공급사슬망관리, 스케줄링, 금융공학

신 현 준(Hyun Joon Shin) **[종신회원]**

- 1995년 2월 : 고려대학교 산업공학과(공학사)
- 1997년 2월 : 고려대학교 산업공학과(공학석사)
- 2002년 2월 : 고려대학교 산업공학과(공학박사)
- 2002년 5월 ~ 2004년 4월 : 미국Texas A&M대학교 산업공학과 Post-Doc
- 004년 6월 ~ 2005년 2월 : ㈜삼성전자 LCD총괄책임연구원
- 005년 3월 ~ 현재 : 상명대학교 경영공학과 조교수

<관심분야>
생산관리, 공급사슬망관리, 스케줄링, 금융공학