2005 한국경영과학회/대한산업공학회 춘계공동학술대회
2005 5 월 13 일~14 일, 충북대학교

Manufacturing/Remanufacturing 환경에서의 유전 알고리즘을 이용한 생산계획

Production Planning in Manufacturing/Remanufacturing Environment using Genetic Algorithm

임치훈
서울대학교 산업공학과
서울시 관악구 신림동 산 56-1

심억수
삼성전자 메카트로닉스 연구소
경기도 수원시 영통구 매탄 3 동 416 번지

박진우
서울대학교 산업공학과
서울시 관악구 신림동 산 56-1

초 록

증가하는 소비의 추세, 한정된 자원, 폐기물 처분 등의 문제로 인하여, remanufacturing 에 대한 관심이 더욱 높아져가고 있다. Remanufacturing 에 대한 생산계획 및 스케줄링은 높은 불확실성으로 인하여 기존의 방법들과 별개로 다루어져야 한다. 본 연구에서는 manufacturing/remanufacturing 생산 시스템에서의 생산계획을 위한 수정된 유전 알고리즘(GA)을 제안하였다. GA 의 chromosome 은 각 부품의 생산 수준을 의미하며, 알고리즘을 이용하여 중간조립품과 최종제품의 분해/조립 계획을 결정할 수 있다. 임의로 생성된 문제들에 대해 실험한 결과, 다양한 문제들에 대하여 적용 가능하다는 것을 알 수 있었다.

Keywords: remanufacturing, aggregate production scheduling, genetic algorithm

1. 서론

자원의 제약과 폐기물로 인한 환경오염 등의 문제들로 인하여 remanufacturing 이 더욱 중요해지고 있다[2]. Remanufacturing 이란 사용된 제품 을 체계적으로 회수, 해체하고 부품을 세척, 수리 또는 새 부품으로 교체해 신제품과 같은 수준 으로 작동할 수 있도록 재조립해 제품화하는 일련의 과정을 말한다. Remanufacturing 제품은 신제품과 같은 수준의 품질과 성능을 보증하는 것은 물론, 그 시점의 최신 기술 수준으로 업그레이드하는 것을 포함하는 것으로 사실상 신제품과 같은 수준으로 복원시키는 것이다.

Remanufacturing 의 production planning and scheduling 은 manufacturing 의 그것과는 구별되어야 한다. remanufacturing 의 가장 큰 특징은 기존 의 manufacturing 보다 현저하게 높은 uncertainty 라고 볼 수 있다. 이를 고려해야 한다는 점에서 기존의 manufacturing 에서 이용되던 production planning 의 방법들을 그대로 적용할 수 없고, 새로운 연구가 진행 되어야 한다. 또한 현실적으로는 하나의 기업이 remanufacturing 만을 하는 경우는 거의 없다. 대부분 manufacturing 과 remanufacturing 을 복합적으로 이용하므로 둘의 통합에 대한 연구가 절실하다.

Manufacturing 과 remanufacturing 의 hybrid manufacturing system 과 관련된 연구는 대부분 재고관리에 집중되어 있었다. Production planning & scheduling 과 관련된 논문은 거의 찾아볼 수 없었다[4]. Clegg 외는 Hybrid manufacturing system 의 production planning and control 에 대하여 linear programming model 을 제시한 바 있다[1]. 제품을 모듈 단위로 한번 분해하고 remanufacturing 을 거치고, 새로운 부품을 외부 조달한 후 조립까지 전 공정을 고려하였으며 capacity 도 고려하였다. 하지만 이 문제를 실제로 해결할 방법에 대해서는 제시하지 못했다.

Taleb 과 Gupta 는 부품 수준의 수요를 고려 하여 여러 제품의 disassembly scheduling 에 대하여 연구하였다[5]. 부품의 수요량을 충족시 키기 위해 회수된 제품을 분해하거나 또는 외부 조달할 경우 분해와 외부조달 비용을 최소화하기 위한 최적 schedule 을 찾는 two-phase algorithm 을 개발했다. 하지만 disassembly shop 의 capacity 를 전혀 고려하지 않는다는 단점이 있다.

본 논문에서는 hybrid manufacturing system 에서의 production scheduling 의 문제를 풀고자 한다. Production scheduling 은 중장기의 sales and operation planning 에서 수요량과 생산용량이 정해진 후, 2~6 주의 단기간에 대하여 제품의 생산 일정을 결정하는 문제이다. 다단계 구조를 가지는 다수의 제품들을 제조 대상으로 한다.

논문은 다음과 같이 구성된다. 먼저 2 장에서 는 본 연구에 다루는 문제를 명확하게 기술하고, 3 장에서는 이 문제의 해결을 위한 알고리즘을 소개한다. 4 장에서 이 알고리즘을 이용한 실험과 결과를 제시하고, 마지막 5 장에서는 결론을 내리고 추후 연구 방향을 제시하고자 한다.

2. Problem Definition

본 연구는 단기간의 hybrid manufacturing 환경에서 production schedule 을 결정하는 것을 목적으로 한다. 그림 1 의 예에서 보듯이 수요량 과 회수량 데이터를 바탕으로 반주(half week) 간격의 production schedule 이 결정된다.

demand	Jan	Feb	Mar	...
Product1	410	350	300	...
Product2	200	230	240	...

recovery	Jan	Feb	Mar	...
Product1	130	40	70	...
Product2	20	50	40	...

period / product		1	2	3	4	5	6	7	8	Sum
1	mfg	20	60	0	0	50	50	50	50	280
	rmfg	30	40	30	30	0	0	0	0	130
2	mfg	0	20	30	50	20	30	30	0	180
	rmfg	0	0	10	10	0	0	0	0	20

그림 1. Production Scheduling 의 예

본 논문에서는 다단계의 BOM(Bill of material) 구조를 가진 다수의 제품을 대상으로 한다. 모든 제품들은 같은 단계의 구조로 가정한 다. 우선적으로 3 단계의 제품들만을 고려하도록 하지만, 이는 이후에 충분히 확장될 수 있다. 그림 2 에서 예시를 볼 수 있다.

생산설비로는 회수된 제품들을 분해하는 disassembly shop 이 필요하며, 분해된 부품들을 remanufacturing 하거나 새롭게 manufacturing 할 수 있는 다수의 workcenter 들이 필요하다. 또한 remanufacturing 이나 manufacturing 을 거쳐 동일한 품질을 가진 부품들을 조립하는 assembly shop 이 필요하다. Disassembly shop 과 assembly shop 은 각각 한 단계씩만 분해하거나 조립할 수 있다고 가정한다. 또한 최하위 부품들은 하나의 특정한 workcenter 에서만 remanufacturing 과 manufacturing 을 모두 끝내고, 여러 부품이 하나의 workcenter 에서 생산될 수 있다고 가정한 다. 3 단계의 제품을 고려하면 다음 그림 2 와 같은 형태의 shop 이 된다.

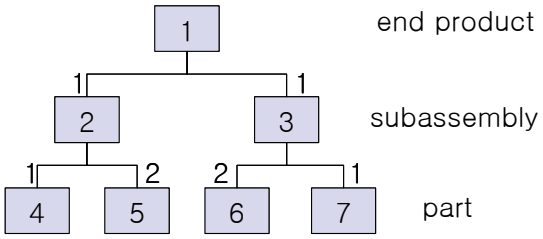


그림 2 제품 구조의 예

목적함수는 전 과정을 거치는 동안의 비용을 최소화하는 것으로 한다. 모든 수요량을 다 생산한다고 가정한다면, 모든 제품을 생산하는 processing cost 는 어떠한 경우에도 같다. 하지만 production scheduling 의 결과에 따라 각 기간별로 생산에 걸리는 시간이 정규노동시간보다 클 경우에는 overtime 생산이 필요하게 되며, 여기에 서 overtime cost 가 추가된다. 또한 어떠한 부품의 manufacturing 또는 remanufacturing 을 시작할 때 setup cost 가 들게 된다. 본 논문에서는 overtime cost 와 setup cost 의 합을 최소화하는 production schedule 를 찾는 것이 목적이 된다.

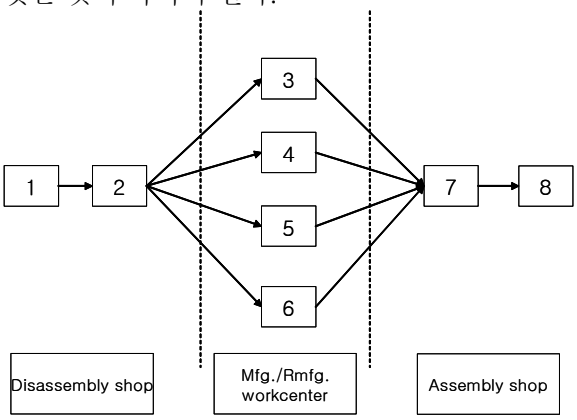


그림 3. 3 단계 구조 제품을 고려한 shop 의 형태

3. Solution Methodology

Scheduling 문제는 NP-complete 문제이기 때문에 genetic algorithm(이하 GA)가 많이 사용되고 있다[3]. 하지만 이 문제에서는 모든 단계의 생산수준들을 GA 의 gene 으로 encoding 할 경우에는, random 하게 생성되는 initial population 이나 crossover, mutation 을 거친 다음 generation 의 population 들이 문제에서 가정하고 있는 조건들을 만족시키지 못하는 경우가 발생할 확률이 높다. Feasibility 를 만족시키지 못한 population 들을 제거하고 새롭게 초기화된 population 으로 대체할 경우, optimal solution 으로의 수렴이 늦어지는 문제가 발생한다.

이러한 문제를 해결하기 위해 본 연구에서는 새로운 알고리즘을 제안한다. 새로운 알고리즘은 부품의 수준마다 스케줄링을 별개로 하는 것이 핵심이다. 물론 수준별로 스케줄링을 따로 한다 고 해서 각 문제 별 최적해만을 찾는 것은 아니다. 각 수준의 스케줄 중 가장 자유롭게 움직일 수 있는 부분은 최하위부품의 manufacturing/ remanufacturing 이라고 볼 수 있다. 이 부분에 대하여 하나의 population 이 생산수준을 결정한 후, 이 계획에 따라 중간조립품이나 최종제품 수준의 생산수준도 결정한다고 볼 수 있다. 전체 계획이 세워지고 나면 전체 cost 가 결정되어 fitness function 값이 결정되고, selection, crossover, mutation 과 같은 과정을 거쳐 최적해를 구하게 된다. 알고리즘의 전체적인 흐름은 그림 4 와 같이 간단히 정리될 수 있다. 이제 각 부분에 대하여 좀더 자세히 설명하도록 한다.

3.1 최하위 부품에 대한 decision variable 생성

먼저 GA 의 chromosome 으로 표현될 decision

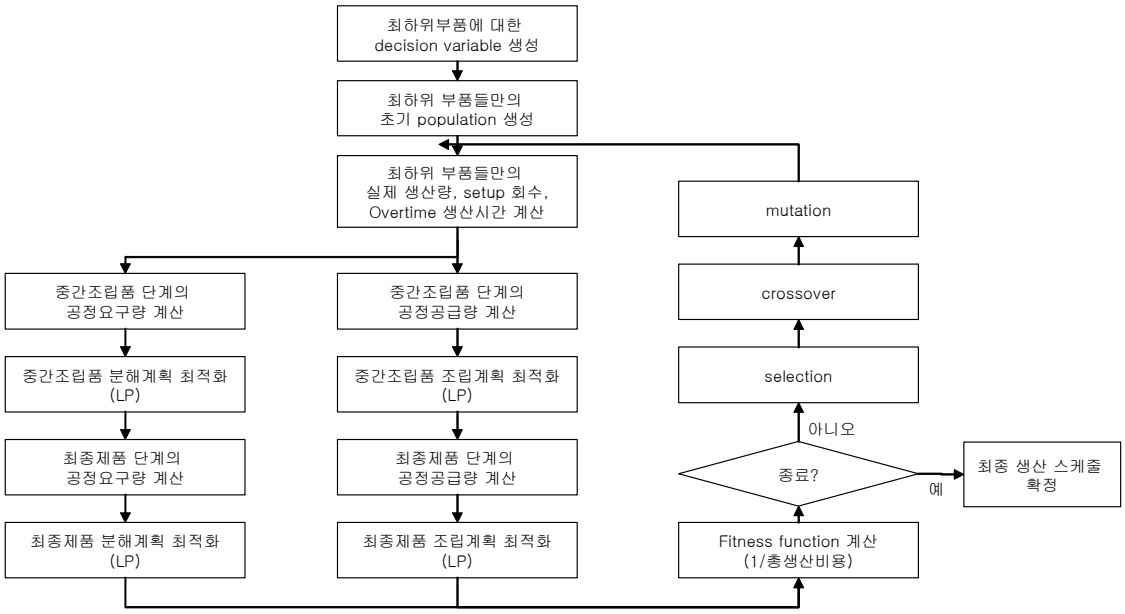


그림 2. Algorithm 의 전반적인 흐름

variable 들을 선택해야 한다. 최하위부품들의 각 기간별 생산수준들이 chromosome 으로 encoding 될 것이다. 하지만 최하위 부품들은 분해와 조립 을 거쳐야 하기 때문에 실제로 생산이 가능한 기간이 전체 기간보다 짧다.

Disassembly 와 assembly 를 위한 기간이 필요하기 때문에 첫 2 기간과 마지막 2 기간에서 는 최하위부품 수준의 remanufacturing 이 불가능 하다. 또한 마지막 2 기간에서는 manufacturing 도 불가능하다.

총 8 기간에 대하여 하나의 제품을 다루는 경우 40 개의 decision variable 이 필요하다. Remanufacturing 은 x_{pt} 로, manufacturing 은 y_{pt} 로 표시한다.

3.2 최하위부품들만의 초기 population 생성

문제에 따라 어떠한 부품과 기간에 대해서 decision variable 을 생성할 것인지가 결정되면, 실제 각 변수들에 대하여 초기 population 을 생성한다. 각각의 부품에 대하여 각 기간에서 생산할 수준을 결정하는 것이다. 이후 실험 계획에서 다시 언급하겠지만, 0/1 로 생산 여부만 결정하는 경우와 0/1/2 로 생산 여부와 함께 생산량에 변화를 주는 경우의 두 가지에 대해서 실험한다. 다음 표 1 은 0/1/2 encoding 으로 생성한 population 의 예이다.

0/1 로 encoding 할 경우에는 feasible solution set 의 크기가 작다는 장점이 있지만, 각 기간에 대하여 생산수준을 달리 할 수 없다는 문제가

표 1 초기 population 의 예

x_{43}	x_{44}	x_{45}	x_{46}	x_{53}	x_{54}	…	y_{76}
0	1	1	2	2	0		2

있다. 반면 0/1/2 로 encoding 할 경우 각 기간의 생산수준을 다르게 하여 optimal solution 에 보다 가깝게 다가갈 수 있지만 feasible solution set 이 0/1 encoding 보다 훨씬 커지게 된다. 3.1 에서 들었던 예의 경우 0/1 encoding 은 2^{40}(≒1012), 0/1/2 는 3^{40}(≒1020) 수준으로 큰 차이를 보인다.

문제의 특성에 따라 이 부분에 대해서 또 다른 형태를 적용할 수 있다. 즉 0/1/2/3 이나 0/3/4 와 같은 형태도 경우에 따라서 적용할 수 있을 것이다.

3.3 최하위부품들만의 실제 생산량, setup 회수, overtime 생산시간 계산

초기 population 이 생성되면 이를 이용하여 최하위부품들의 실제 생산량을 구할 수 있고, setup 회수, overtime 생산시간도 계산할 수 있다.

표 1 의 예에서 부품 4 의 remanufacturing 양이 100 일 경우,

$$x_{44} = \frac{1}{0+1+1+2} \times 100 = 25$$

와 같은 방법으로 계산된다. 그러므로 한 부품에 대하여 모든 기간의 생산수준이 0 으로 표현되지 만 않으면 feasibility 가 유지된다. 만약 모든 기간에 대하여 0 으로 표현되는 경우에는 그 부품에 대해서만 초기화를 다시 실시한다.

또한 setup cost 를 구하기 위해서 각 부품의 setup 회수를 계산해야 한다. 이전 기간에 생산이 없고, 현재 기간에 생산이 발생할 경우에 setup 이 한번씩 발생하는 것으로 계산한다. 부품 4 의 remanufacturing 과 관련 하여 다음과 같이 계산된다.

SU_p=0, x_{p2}=0
for i=3 to 6
if $x_{p(i-1)}$=0 and x_{pi}>0 then
SU_p=SU_p+1
end if
end

Overtime 생산시간은 각 workcenter 에 대하여 기간별로 계산된다. 각 workcenter 에서 작업되는 부품들의 processing 시간의 합이 workcenter 의 capacity 를 초과할 경우 overtime 생산시간으로 계산되게 된다. 다음과 같은 방법을 이용한다.

if $c_{st} \geq \sum_p x_{pt} rpt_p + \sum_p y_{pt} pt_p$

then $ot_{st} = 0$

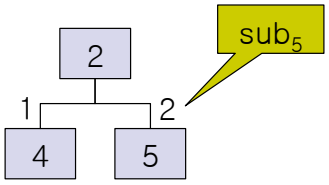
else $ot_{st} = \sum_p x_{pt} rpt_p + \sum_p y_{pt} pt_p - c_{st}$

end

c_{st} : capacity of workcenter s for period t
ot_{st} : overtime working hours of workcenter s for period t
rpt_p : remanufacturing processing time for part p
pt_p : manufacturing Processing time for part p

3.4 중간조립품 단계의 공정요구량 계산

최하위 부품들에 대한 remanufacturing schedule 이 구해지면 이를 이용하여 상위 단계의 subassembly 의 분해 schedule 의 공정요구량(process requirement)을 구할 수 있다. 공정요구량은 본 연구의 문제를 해결하기 위해 새롭게 도입한 개념이다. 즉 period i 에서 부품 p 를 x 개 remanufacturing 하기 위해서는 period 1 부터 period (i-1)까지 이미 상위 부품이 그만큼 분해되어 있어야 한다는 의미이다. 다음 그림 5 에서 예를 들어보면 부품 4, 5 에 대한 schedule 이 주어지면, 부품 2 의 공정요구량을



part \ period	2	3	4	5
PR of 2	60	60	20	0
4 rmfg		60	50	30
5 rmfg		100	140	40

그림 5 공정요구량 계산의 예

계산한다. 기간 2 에서는 부품 4 로 인하여 공정요구량이 정해진다. 이는 기간 1 과 2 동안 부품 2 가 60 개 이상 분해되어야 한다는 것을 의미한다. 기간 3 의 공정요구량은 부품 5 로 인하여 결정된다. 부품 5 가 기간 4 까지 240 개 remanufacturing 되어야 하므로 이전까지 부품 2 가 120 개 분해되어 있어야 한다. 부품 2 는 기간 2 까지 60 개가 분해된다고 했으므로 기간 3 에서는 60 개가 더 분해되어 있어야 한다.
이러한 공정요구량의 계산 방법을 식으로 나타내면 다음과 같다.

$$pr_{pt} = \max_p \left\{ \frac{\sum_{i=1}^{t+1} x_{pi}}{sub_p} \right\} - \sum_{j=1}^{t-1} pr_{pj}$$

3.5 중간조립품 분해계획 최적화

공정요구량들이 결정되면 중간조립품들의 분해계획은 쉽게 구해질 수 있다. 즉 그림 5 에서처럼 공정요구량이 결정되고 부품 2 의 각 기간별 분해량을 x_{22}, x_{23}, x_{24} 라고 두면,

$x_{22} \geq 60$
$x_{22} + x_{23} \geq 60 + 60$
$x_{22} + x_{23} + x_{24} \geq 60 + 60 + 20$

와 같은 형태의 제약식을 생성할 수 있다. 또한 전체 분해량이 부품 2 의 전체 분해량과 같아야 한다는 제약식이 필요하고, 목적함수는 overtime cost 를 최소화하는 것이다. 이는 linear programming(이하 LP)방법으로 쉽게 해결할 수
있다. 중간조립품들에 대해서만 LP 를 사용할 경우에는 변수의 수가 그리 크지 않아서 계산시간은 크게 오래 걸리지 않는다.

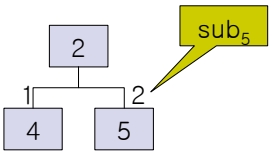
3.1 에서 들었던 예의 경우, 부품 2 와 3 을 고려하고 기간은 4 개 기간만 고려하면 된다. 다만 목적함수의 형태가 capacity 를 초과할 경우에만 발생하는 overtime cost 의 형태이기 때문에 LP 를 이용하려면 각각의 변수를 모두 2 개로 나누어야 하므로, 전체 변수의 수는 16 개면 된다.

3.6 최종제품 단계의 공정요구량 계산 및 분해계획 최적화

3.5 에서 중간조립품들의 분해계획이 생성되면, 이를 이용하여 마찬가지의 방법으로 공정요구량을 구하고 최적화된 분해계획을 생성할 수 있다.

3.7 중간조립품 단계의 공정공급량 계산

중간조립품의 조립계획도 분해계획과 유사하게 공정공급량(process supply)이라는 개념을 이용하여 설명할 수 있다. 즉 period i 에서 부품 p 를 x 개 조립하기 위해서는 period 1 부터 period (i-1)까지 이미 하위 부품 모두가 그만큼 manufacturing 이나 remanufacturing 되어 있어야 한다는 의미이다. 아래 그림 6 에서처럼 기간 4 에서 부품 2 를 조립하기 위해서는 기간 3 까지 manufacturing/remanufacturing 된 부품 4 와 5 가 필요하다. 부품 5 는 80 개가 생산되어 부품 2 를 40 개 조립하도록 준비되어 있지만, 부품 4 의 경우 30 개 밖에 생산되지 않았기 때문에 부품 2 는 현재 30 개만 생산할 수 있는 상태이다. 기간 5 까지는 부품 4 는 110 개까지 조립할 수 있지만, 부품 5 가 60 개 밖에 조립할 수 없기 때문에 기간 4 까지의 공정공급량 30 을 제외하고 기간 5 의 공정공급량은 30 이 되는 것이다.



part \ period	3	4	5	6	7
PP of 2		30	30	50	0
4(mfg+ rmfg)	30	80	0		
5(mfg+ rmfg)	80	40	100		

그림 6 공정공급량 계산의 예

이러한 공정요구량의 계산 방법을 식으로 나타내면 다음과 같다.

$$ps_{pt} = \min_{p}\left\{\frac{\sum_{i=1}^{t-1}(x_{pi} + y_{pi})}{sub_p}\right\} - \sum_{j=1}^{t-1} ps_{pj}$$

3.8 중간조립품 조립계획 최적화

공정공급량들이 결정되면 중간조립품들의 조립계획은 쉽게 구해질 수 있다. 즉 그림 6 에서처럼 공정요구량이 결정되고 부품 2 의 각 기간별 조립량을 y_{24}, y_{25}, y_{26}, y_{27} 으로 두면,

$y_{24} \le 30$

$y_{24} + y_{25} \le 30 + 30$

$y_{24} + y_{25} + y_{26} \le 30 + 30 + 50$

와 같은 형태의 제약식을 생성할 수 있다. 또한 전체 조립량이 부품 2 의 전체 조립량과 같아야 한다는 제약식이 필요하고, 목적함수는 overtime cost 를 최소화하는 것이다.

분해 계획과 마찬가지로 변수의 수가 그리 크지 않아서 LP 를 이용하더라도 계산시간은 크게 오래 걸리지 않는다. 4.1 의 예에서는 부품 2, 3 을 고려하고 총 6 기간을 고려하고, 각 변수를 2 개로 나누기 때문에 24 개의 변수를 가진다.

3.9 최종제품 단계의 공정공급량 계산 및 조립계획 최적화

3.8 에서 중간조립품들의 조립계획이 생성되면, 이를 이용하여 마찬가지의 방법으로 공정공급량을 구하고 최적화된 조립계획을 생성할 수 있다.

3.10 Fitness function 계산

모든 단계별로 schedule 이 결정되면 총생산비용을 계산할 수 있다. 2 장에서 이미 언급한 것처럼 총생산비용은 최하위부품들의 setup cost 와 각 workcenter 별 overtime cost 로 구성된다.

Setup cost 는 3.3 에서 계산된 setup 횟수와 setup 1 회 비용의 곱으로 계산된다. Overtime cost 는 workcenter 별로 이미 계산된 overtime 생산시간을 이용한다. 최하위부품들이 manufacturing/remanufacturing 되는 workcenter 3, 4, 5, 6 의 overtime 은 3.3 에서 이미 계산되었으며, 조립이나 분해가 이루어지는 workenter 1, 2, 7, 8 의 overtime 생산시간은 3.5, 3.6, 3.8, 3.9 에서 LP 를 계산한 목적함수값을 사용하면 된다.

GA 에서 각각의 population 의 적합성을 판단하는 Fitness function 값은 높을수록 이후의 유전이 진행될 때 선택될 확률이 높아진다. 본 연구의 문제는 최소비용을 찾는 문제이므로 fitness function 으로 총생산비용을 그대로 사용할 수 없다. 그러므로 본 연구에서는 다음과 같은 fitness function 을 사용한다.

fitness function = 1/총생산비용)=1/(최하위부품들의 setup cost+ workcenter 별 overtime cost)

3.11 알고리즘 종료 여부 결정

모든 population 의 fitness function 이 계산되면 알고리즘의 종료 여부를 결정한다. 종료 조건은 현재 generation 의 fitness function 의 평균값이 최초 generation 의 fitness function 평균값의 2 배 이상 되거나, 총 generation 수가 100 을 넘는 것이다. Fitness function 평균값을 기준으로 결정한 이유는, 전반적으로 진화가 잘 이루어지고 있다는 것을 알 수 있는 수치이기 때문이다.

종료 조건을 만족시킬 경우, 가장 좋은 fitness 값과 그 population 을 출력하고 종료한다. 만족시키지 못할 경우에는 3.12 의 selection, crossover, mutation 을 실시한다.

3.12 Selection, Crossover, Mutation

Selection, crossover, mutation 은 현재 generation 의 population 들을 유전시켜 다음 generation 의 population 들을 생성하는 것이다.

먼저 selection 은 이전 부모 generation 의 population 들 중 fitness 값이 높은 population 들을 선택하는 과정이다. Normalize 된 fitness 값이 population 의 생존 확률이 된다.

생존한 population 들로부터 child 를 생성하는 것이 crossover 과정이다. 아래 그림 7 에서처럼 두 개의 부모 population 에서 먼저 난수 생성을 통해 2 개의 crossover point 를 선택한다. 선택된 두 지점 사이에서 부모 간의 gene 이 서로 바뀌게 된다. Selection 을 거쳐 fitness 가 높은 부모 population 이 선택되었기 때문에 이 과정을 통하여 보다 fitness 가 높은 child population 이 나올 수도 있다.

전체 population 들 중 crossover 될 확률을 crossover rate(이하 Pc)라고 하는데, 이에 대해서는 이후의 실험을 통해 0.3 과 0.1 로 놓고 결과를 비교해 본다.

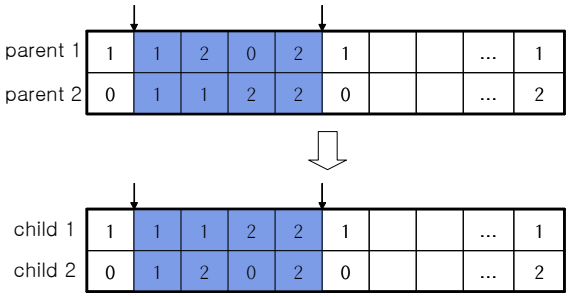


그림 7 Crossover 의 예

Crossover 를 마치면 mutation 을 일으킨다. 단순히 crossover 만을 거칠 경우 전체 해들이 local optimum 에 수렴하여 global search 를 계속 진행할 수 없기 때문이다. Mutation 은 각 population 에서 random 하게 선택된 gene 에 대하여 이루어진다. 0/1 encoding 의 경우는 선택된 gene 이 0 이면 1 로, 1 이면 0 으로 서로 반대로 바꾼다. 0/1/2 encoding 의 경우는 선택된 gene 이 0 이면 1 로, 1 이면 2 로, 2 이면 0 으로 바꾼다.

Mutation 되는 gene 의 비율을 mutation rate(이하 Pm)라 하는데, 이후 실험에서는 0.01 과 0.05 의 두 가지 경우에 대하여 결과를 비교해 본다.

이러한 세 가지 과정이 끝나면 새로운 generation 이 생성되며, 3.3 부터의 과정을 다시 반복하게 된다.

4. Computational Experiments

본 연구는 현재까지 해를 구하지 못했던 문제를 해결하고자 하는 첫 시도이다. 그러므로 다른 연구와의 실험 결과를 비교할 방법은 없다. 다만 본 연구의 알고리즘의 fitness 값이 generation 이 진행됨에 따라 증가하는지, encoding 방법이나 crossover rate, mutation rate 등을 변화시키면서 어떠한 setting 이 문제를 푸는 데 보다 적절한 것인지에 대하여 실험해본다. 또한 그러한 setting 이 하나의 문제에 대해서만 유용한 것이 아니라, 데이터가 다른 여러 가지 문제 set 에 대해서도 잘 작동하는지에 대해 실험한다.

실험은 크게 두 부분으로 나누어진다. 첫 번째 실험은 하나의 문제 set 에 대하여 가장 좋은 결과를 나타내는 최적의 setting 조합을 찾아내는 것이다. 두 번째 실험은 첫 번째 실험에서 찾은 최적의 setting 조합으로 다른 문제 set 들에 대해서도 적용할 수 있는지를 알아보는 것이다.

4.1 에서 우선 실험에 필요한 문제 set data 를 어떻게 생성하였는지에 대하여 설명하고, 4.2 에서 각각의 실험에 대해서 자세히 설명한다.

실험에서 사용되는 문제 set 들은 각각의 변수별로 특정한 분포를 따르는 난수를 생성해서 사용한다. 2 종류의 제품을 대상으로 하며, 총 8 기간 동안의 production scheduling 을 실시한다.

Workcenter 의 배정은 다음 표 2 와 같이 선택되었다. 각 문제에 따라 3 개 이상의 부품이 하나의 workcenter 에 집중되는 경우(노란색)도 있고, 심지어 4 개, 5 개의 부품이 하나의 workcenter 에 집중되는 경우(붉은색)도 있었다. 문제 set 8 의 경우 5 개의 부품이 한 곳에, 나머지 3 개의 부품이 다른 한 곳에 집중되는 경우도 있었다. 이와는 반대로 문제 set 5 의 경우에는 모든 workcenter 에 2 개의 부품씩 배정되어 고른 분포를 나타내는 경우도 있었다.

문제들이 이렇게 다양한 형태로 나타나기 때문에, 알고리즘을 여러 가지 형태의 문제 해결에 시도해 볼 수 있다.

표 2 각 workcenter 별 부품 할당

part / problem	4	5	6	7	11	12	13	14
1	6	4	4	4	6	5	4	5
2	5	6	5	6	4	4	6	4
3	4	6	6	6	3	3	4	5
4	6	4	6	4	5	5	6	4
5	3	6	5	5	4	3	4	6
6	5	5	4	6	5	3	4	5
7	6	5	6	4	6	4	4	4
8	4	4	4	4	5	4	5	5
9	5	6	4	4	4	3	5	4
10	3	6	5	6	3	5	3	6

4.1 실험 1

첫 번째 실험은 문제 set 1 에 대해서만 GA 의 setting 을 변화시키면서 최적의 setting 조합을 찾는 실험이다. Setting 은 3 장에서 이미 언급한 것처럼, chromosome encoding, crossover rate, mutation rate 에 대하여 실험한다. 각각의 setting 방법은 다음과 같다.

- Chromosome encoding 방법: 0-1, 0-1-2
- Crossover rate: 0.1, 0.3
- Mutation rate: 0.01, 0.05

이러한 3 개의 setting 방법에 대하여 8 개의 실험군이 생기고, 각각에 대해 실험을 실시하여 최대 fitness 값과 각 generation 별 최대 fitness 값의 추세 등을 보고 최적의 조합을 선택한다.

각각의 실험군의 실험결과는 다음 표 3 과 같다. ‘ 최대 fit gen index’ 는 fitness 가 가장 높은 값이 나왔던 generation 의 번호를 의미한다. ‘ 1th gen 최대 fit(a)’ 은 첫 번째 generation 의 population 중 최대 fitness 값을 의미하며, ‘ 전체 최대 fit(b)’ 은 알고리즘 종료까지 최대로 높게 나온 fitness 값을 의미한다. Fitness 값은 1/총생산비용 이므로 ‘ a/b’ 은 generation 이 진행됨에 따라 최소비용이 얼마나 감소했는지를 의미한다고 볼 수 있다.

먼저 최대 fitness 값(b)의 측면에서 볼 때, 실험군 3 과 6 이 0.5652 로서 가장 높은 값을 보인 것으로 나타났다. 하지만 각 generation 별 fitness 평균을 고려하면, 실험군 3 과 6 은 다른 양상을 나타내었다. 실험군 3 의 경우는 generation 이 진행됨에 따라 평균이 점차 증가하는 추세를 보임으로서, population 들이 전반적으로 진화하고 있다는 것을 알

수 있었다. 반면에 실험군 6 의 경우 generation 이 진행되어도 fitness 평균이 증가하지 않는 것으로 나타나, 전반적으로 진화가 일어났다고 보기 힘들고, random search 에 가까운 결과가 나왔다는 것을 알 수 있었다.

표 3 실험 1 의 결과

encoding	crossover rate	mutation rate	max fit gen index	1st gen max fit(a)	max fit(b)	a/b	min cost
01	0.1	0.01	79	0.5331	0.5425	0.983	18433
01	0.1	0.05	1	0.5652	0.5652	1.000	17694
01	0.3	0.01	20	0.5374	0.5487	0.979	18224
01	0.3	0.05	1	0.5342	0.5342	1.000	18720
012	0.1	0.01	31	0.4648	0.5342	0.870	18720
012	0.1	0.05	97	0.5161	0.5508	0.937	18155
012	0.3	0.01	92	0.4768	0.5652	0.844	17694
012	0.3	0.05	48	0.5007	0.5433	0.855	18406

0/1 encoding 을 사용한 실험군에서 이러한 결과를 많이 찾아볼 수 있었다. 실험군 6, 7, 8 의 경우 최대 fitness 값이 generation 이 진행된 지 얼마 되지 않아 나타난다는 것을 알 수 있다. 최대 fitness 값이 앞쪽에서 이미 나오지만, 그로부터 더 이상의 해의 진화가 이루어지지 않는다. 이러한 문제로 0/1 encoding 보다 0/1/2 encoding 이 이러한 문제를 해결하는 데 있어서 보다 나은 결과를 나타낸다.

이는 생산 수준의 구별에서 오는 결과이다. 0/1 encoding 에서는 각 기간에서의 생산 여부만이 결정될 뿐, 생산이 이루어지는 기간들 사이에서는 생산수준이 모두 같다. 하지만 0/1/2 encoding 에서는 생산 여부뿐만 아니라 생산수준도 결정되기 때문에 보다 최적해에 가깝게 다가갈 수 있다는 것이 장점이 된다. 다만 알고리즘 소개에서도 밝힌 바와 같이 feasible solution set 이 훨씬 방대해지기 때문에, 해를 찾는 시간이 좀더 오래 걸릴 수 있다. 또한 population 의 수가 더 커져야 빠르게 해를 찾을 수 있을 것으로 보인다.

실험군 3 의 결과를 다시 보면 crossover rate 는 0.3 이 0.1 보다 좋은 결과를 나타냈다. 다른 조건이 모두 같을 경우 0.3 인 경우가 ‘ a/b’ 의 값이 더 낮게 나온다. 0.1 의 경우에는 crossover 되는 population 의 비율이 너무 적어서, 다른 해로 나아가는 데 시간이 더 많이 걸리는 것으로 보인다.

Mutation rate 도 다른 조건이 모두 같을 때, 0.01 인 경우가 ‘ a/b’ 값이 더 낮게 나오는 것을 볼 수 있다. Mutation rate 가 0.05 로 높은 경우에는 좋은 해들조차 변이를 일으키기 때문에 보다 좋은 해를 찾는 것이 어려워진다고 분석된다.

이러한 결과를 종합하면 최적의 조합은 0/1/2 encoding, crossover rate 0.3, mutation rate 0.01 이라고 볼 수 있다. 이 결과를 이용하여 실험 2 를 시행하였다.

4.2 실험 2

첫 번째 실험에서 최적의 조합이 선택되면, 이 조합으로 다른 문제 set 들도 해결할 수 있을지에 대하여 실험한다. 같은 조합으로 각각의 문제 set 에 적용시키고, 마찬가지로 최대 fitness 값과 각 generation 별 최대 fitness 값의 추세 등을 살펴본다.

실험 1 에서 찾은 최적의 조합을 나머지 문제 set 9 에도 적용해보았을 때의 결과를 다음 표 4 와 같이 정리할 수 있다. 실험 1 과 마찬가지로 ‘ a/b’ 값과 최대 fitness 값(최소비용)의 두 가지 측면에서 결과를 분석해본다.

먼저 ‘ a/b’ 값을 보면 문제 set 9 를 제외한 대부분의 문제 set 에서 첫 번째 generation 보다 fitness 값이 많이 증가하였다는 것을 볼 수 있다. 문제 set 1 뿐만 아니라 여러 가지 형태의 문제에서도 알고리즘이 대체적으로 잘 작동한다는 것을 의미한다. 각 generation 별 평균값의 변화를 살펴본 결과도, 모든 문제 set 에서 generation 이 진행됨에 따라 평균값이 1.5 배 이상 증가함을 볼 수 있었다. 하지만 종료 조건으로 설정한 2 배까지는 증가하는 경우가 없었다. 그래서 모든 문제 set 들에 대하여 100 generation 까지 실험이 계속되었다.

set 7 의 경우에는 첫 번째 generation 에 비해 최소비용이 70% 이하까지 감소했다. 반면에 set 9 의 경우에는 첫 번째 generation 에서 최대 fitness 값이 나왔고, 그 이후로 더 높은 fitness 값이 나오지 않았다. 물론 이 경우에도 fitness 의 평균값은 계속 증가하는 추세를 보여 알고리즘을 따라 진화가 진행되었음은 확인할 수 있었다. 다만 첫 번째 generation 에서 random 하게 생성된 population 중 아주 좋은 결과를 나타내는 population 이 우연히 나오게 된 것으로 판단된다.

표 4 실험 2 의 결과

problem	max fit gen index	1st gen max fit(a)	max fit(b)	a/b	min cost
1	92	0.4768	0.5652	0.844	17694
2	100	0.6455	0.8610	0.750	11614
3	91	0.6206	0.8214	0.756	12174
4	84	0.6732	0.8489	0.793	11780
5	48	0.9506	1.1390	0.835	8780
6	63	0.5175	0.6266	0.826	15959
7	49	0.6689	0.9613	0.696	10403
8	97	0.5720	0.7030	0.814	14225
9	1	0.6530	0.6530	1.000	15315
10	54	0.6029	0.7946	0.759	12585

최대 fitness 의 측면에서 실험의 결과를 분석해본다. 최대 fitness 를 이용하여 최소비용(=1/fitness)을 구할 수 있다. 각각의 문제 특성에 따라 최소비용에서 차이가 난다. 문제 set 5 의 경우 가장 낮은 최소비용을 나타내는데, 이는 부품들이 workcenter 들에 고르게 배정되어 있기 때문이다. 각 workcenter 에서의 작업량이 적기 때문에, overtime cost 가 감소할 것이고, 짧은 기간 내에 하나의 부품에 대하여 생산을 마칠 수 있기 때문에 setup cost 도 감소하여 이와 같은 결과를 가져왔다. 반면 set 1 의 경우에는 workcenter 4 에 4 개의 부품이 집중되어

있으며, 수요량도 다른 문제 set 에 비해 높게 설정되어 있어서 비용이 가장 높은 것으로 분석되었다.

5. Conclusion

본 연구는 manufacturing 과 remanufacturing 을 동시에 고려한 aggregate production scheduling 문제를 풀기 위한 알고리즘을 제시하였다는 점에 의의를 둘 수 있다. 기존의 연구에서 수학적 모델링만을 제시하고, 해법을 제시하지 못했던 것에서 발전된 것이다. 알고리즘을 통해 반드시 최적해를 구한다고 할 수는 없지만, 현재까지 해를 구하는 방법이 존재하지 않았기 때문에 첫 시도로서 의미를 가진다.

다단계 구조를 가지는 다수 제품을 다수 workcenter 에서 생산하는 복잡한 형태의 문제를 해결하기 위해 genetic algorithm 을 사용하였다. 모든 단계의 부품들에 대하여 GA 의 chromosome 을 생성할 경우, feasible solution set 이 너무 커지고, solution 이 infeasible 한 경우가 많아진다는 단점이 있다. 이러한 단점을 보완하기 위해 가장 자유롭게 움직일 수 있는 최하위부품들의 생산수준에 대해서만 chromosome 을 생성하였다. 이외의 상위 부품들의 scheduling 에는 linear programming 을 이용하여 해결하였다.

1 개의 문제 set 에 대해서 setting 을 변화시키면서 실험을 실시한 결과, 0/1/2 encoding, crossover rate 0.3, mutation rate 0.01 이 최적의 조합임을 알 수 있었다. 그리고 이 조합을 다른 9 개의 문제 set 에 적용시켜 본 결과 전반적으로 좋은 결과를 나타내었다.

하지만 본 연구에서도 아직 부족하여 추후에 연구가 더 필요한 부분이 있다. 먼저 알고리즘이 scale 이 큰 문제에 대해서는 좋은 결과를 나타내지 못했다는 점이다. 제품의 수가 더 많은 문제들에 대해서는 진화가 잘 되지 않는 문제가 있었다. 이를 해결할 방법에 대해서 추가적인 연구가 필요하다.

6. REFERENCES

[1] Clegg, A.J., Williams, D.J., Uzsoy, R., Production planning for companies with remanufacturing capacity. Proceedings of the 1995 IEEE International Symposium on Electronics and the Environment, Orlando, FL, 186-191

[2] Fleischmann, M., Quantitative models for reverse logistics. Springer-Verlag, Berlin, 2001

[3] Glibovets, N.N., Medvid, S.A., Genetic algorithms used to solve scheduling problems. Cybernetics and Systems Analysis, 2003, 39(1), 81-90

[4] Guide, V.D.R. Jr., Production planning and control for remanufacturing: industry practice and research needs. Journal of Operations Management, 2000, 18, 467-483

[5] Taleb, K.N., Gupta, S.M., Disassembly of multiple product structure. Computers and Industrial Engineering, 1997, 32(4), 949-961