Journal of the Society of Korea Industrial and Systems Engineering
Vol. 32, No. 4, pp.192－202, December 2009.

불확실성 하에서 최대후회 최소화 분해 계획

강 준 규†

성결대학교 산업경영공학부

Minmax Regret Approach to Disassembly Sequence Planning with Interval Data

Jun-Gyu Kang†

Department of Industrial Engineering, Sungkyul University

Disassembly of products at their end-of-life (EOL) is a prerequisite for recycling or remanufacturing, since most products should be disassembled before being recycled or remanufactured as secondary parts or materials. In disassembly sequence planning of EOL products, considered are the uncertainty issues, i.e., defective parts or joints in an incoming product, disassembly damage, and imprecise net profits and costs. The paper deals with the problem of determining the disassembly level and corresponding sequence, with the objective of maximizing the overall profit under uncertainties in disassembly cost and/or revenue. The solution is represented as the longest path on a directed acyclic graph where parameter (arc length) uncertainties are modeled in the form of intervals. And, a heuristic algorithm is developed to find a path with the minimum worst case regret, since the problem is NP-hard. Computational experiments are carried out to show the performance of the proposed algorithm compared with the mixed integer programming model and Conde's heuristic algorithm.

Keywords : Assembly/Disassembly Sequence, Minmax Regret, Robust Shortest Path, Interval Data

1. 서 론

환경 문제가 중요한 이슈가 되면서 수명이 끝난 제품들과 산업폐기물의 처리는 이미 우리 기업과 사회에서 처리를 늦출 수 없는 문제가 되고 있다. 우리나라에서도 2003년부터 확대생산자책임제도(EPR, Extended Producer Responsibility)를 도입하여 소비자가 사용하던 제품이 폐기물로 배출될 때 생산자에게 일정량을 회수하도록 의무를 부여하고 있으며, 특히 복합재료로 이뤄진 전자-기계(electro-mechanical) 제품들의 경우는 소각이나 매립 등이 점차 금지되는 추세로, 2008년 1월부터는 EPR 제도에서 '전기 전자제품 및 자동차의 자원순환에 관한 법률'을 제정하여 시행하고 있다[22]. 일례로, 1980년대 고안된 버려진 폐기물에서 필요한 자원을 재활용하자는 도시광산(Urban Mining)이라는 개념이 일본에서부터 최근에 각광을 받기 시작하는 것을 보더라도, 제한된 자원 문제를 해결하는 유일한 대안은 보다 많은 부품과 제품들을 재활용 또는 재사용하는 것뿐일 것이다[1].

파쇄에 의한 소재의 재활용만을 고려하는 경우에는, 휴대폰에서 1kg의 금을 추출하기 위해 17ton의 폐기물이 발생하는 것처럼, 리사이클링 과정에서 따로 처리해야 하는 폐기물 양이 많아지므로 제품의 분해공정은 재사용 및 재활용 공정상에서 선결요소이다. 그러나 부품 수가 2만 개에 이르는 자동차와 같이 복잡한 구성 요소와 복

논문접수일 : 2009년 10월 10일 논문수정일 : 2009년 12월 15일 게재확정일 : 2009년 12월 18일
† 교신저자 kangjungyu@hotmail.com

합적인 재료로 이루어진 제품들의 경우 재사용/재활용 과정에서 분리 및 부분적 회수가 어려운 실정이다[29]. 일반적으로 순수 소재의 회수 추출, 유해 물질의 제거, 또는 재사용 가능한 부품 또는 부품군의 분리 등의 목적으로 적용되는데, 이런 분해공정을 효과적으로 수행하기 위한 일련의 과정을 분해공정계획(Disassembly Process Planning)이라 한다. 분해공정계획은 제품구조표현과 분해순서계획을 포함하고, 분해순서계획은 다시 분해순서 생성과 최적분해순서 결정으로 나누어지는데, 이 세 과정은 동시에 혹은 순차적으로 수행될 수 있으나 서로 밀접한 연관성을 가진다[20].

초기의 분해순서계획에 대한 연구는 조립은 분해의 역순이라는 관점에서 자동화된 조립계획(Assembly Plan)을 시작되었는데[13], 분해공정은 일반적으로 2가지 측면에서 조립공정과 다르다. 첫째, 분해는 조립과 달리 선택적(selective disassembly)으로 이루어진다. 즉, 초기 상태와 최종상태가 정해진 조립과는 달리, 일반적으로 분해공정에서는 경제적 혹은 환경적 영향을 고려하여 불완전한(incomplete) 분해를 수행하게 되므로, 분해순서뿐만 아니라 분해 깊이(disassembly level 또는 disassembly depth)와 이에 따른 제품의 최종상태를 결정해야 하고, 이 때 추출되는 부품의 가치에 기반을 둔 경제적 의사결정이 요구된다[15, 30, 31].

둘째, 분해 과정에는 불확실성(uncertainty)이 크다[17]. 분해공정에는 부품과 체결부위의 결함, 분해상의 손상 등에 의한 분해시간의 변동이 크고, 추출된 부품 혹은 소재의 가치(recovery value)의 변동 역시 크다. 그런데, 이와 같은 분해의 불확실성은 기본적으로 정보의 부족에 기인하고, 분해 대상이 반복 수행이 아니라 일회성에 그치는 경우가 많아 확률적 모형화가 어려우며, 분해계획을 위한 선형계획법에 기반을 둔 민감도분석 역시 분해공정 전반에 걸쳐 불확실성이 크게 나타나는 경우 적용하기 어렵다[12, 23]. 확률 이론에 기반을 두지 않는 퍼지집합(fuzzy set)으로 모호한(imprecise) 데이터를 모형화 할 수 있는데[24], Chevron et al.[9]는 모호한 작업시간을 Fuzzy number로 모형화한 후 일반 선형계획법(crisp linear Programming)으로 변환하였고, Tripathi et al.[32]은 fuzzy number로 모수(parameter)를 모형화하고 개미집단최적화 기법을 적용하여 분해순서를 결정하고자 하였다.

위에서 언급한 2가지 분해순서계획의 특성을 고려할 때, 작업시간이 순서 의존적인 선택적 분해순서 결정문제는 순환이 없고 방향이 주어진 네트워크(*directed acyclic graph*, 이하 DAG)에서 최장경로문제로 모형화 될 수 있고, 모수의 모호성(imprecision)은 그 실현가능성 범위를 간격(interval data)로 모형화 할 수 있는데, 이렇게 불확실성이 있는 경우에 적절한 의사결정 기준으로 최대후회 최소화 기준(minimax regret criterion)을 적용할 수 있다[16, 17]. 최대후회는 달리 최악의 경우 후회(worst-case regret) 또는 경영학에서 기회손실(opportunity loss) 등으로 해석된다. 이렇게 분해순서계획 문제로부터 변환된 수리적 문제를 DAG*상에서 가지(arc)길이의 간격이 주어진 최대후회 최소화 최장경로문제(the Minmax Regret Longest Path problem on directed acyclic graph with interval data,*이하 간단히 MRLP라 함)라 한다.

Karaşan et al.[21]은 가지 길이의 간격이 주어진 최대후회 최소화 최단경로문제(DAG와 무관, 이하 MRSP라 함)에 대한 혼합정수계획모형(Mixed Integer Linear Programming, 이하 MILP)을 제시하고, dominated arc(주경로 상에 존재할 가능성이 없는 가지)를 판별하기 위한 전처리기법을 제안하였다. Karaşan et al.[21]의 MILP 모형에 기초하여 Montemanni et al.[27]은 수행시간을 단축할 수 있는 분단탐색법을 제안했고, 이어서 Karaşan의 MILP 모형에 Benders decomposition approach를 이용하여 분단탐색법의 수행속도를 향상시키고자 하였다[28]. 반면에, Kang[18]은 동적계획법 모형에 기반을 둔 최적해 탐색 알고리듬을 제안하였다.

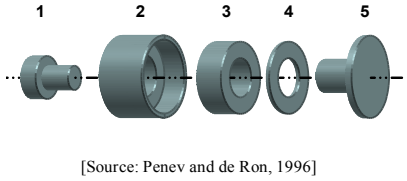
그러나 MRLP와 MRSP는 모두 전형적인 *NP*-hard문제로 알려져 있으므로[4, 34], 현실적인 접근법으로써 Montemanni and Gambardella[26]는 순위 기반의 휴리스틱 알고리듬을 제안하였는데, Eppstein[11]의 알고리듬을 이용하여 k-최단경로(최단경로부터 k번째 최단경로까지)를 찾고 그 중 최대후회가 최소인 경로를 선택한다. Kang et al.[17] 역시 유사한 방법을 제안하였는데, *k*-최단경로를 생성하기 위해 branch-and-cut 방법을 이용한다. 그러나 핵심인 k개의 최적해가 될 가능성이 있는 경로를 생성하는 문제 역시 NP-hard이므로 대형 MLRP 또는 MLSP에서 효과적일 것으로 기대할 수 없다[7, 8]. 반면에, Conde[10]는 동적계획법 모형을 제안하고, 제안한 동적계획법모형을 단순화하여 대단히 빠르게 수행되는 휴리스틱 알고리듬을 제안하였으나 탐색해의 정확도가 낮다. 그러므로 본 연구는 분해순서계획을 위해 MRLP의 정확도가 높은 해를 효과적으로 찾는 휴리스틱 알고리듬을 제안하고자 한다.

본 논문은 구성은 다음과 같다. 제 2장은 최대후회 최소화 최장경로문제에 대한 설명 및 Karaşan et al.[21]의 MILP 모형을 설명하고, 제 3장에서는 Kang[18]의 동적계획법 모형에 기반을 둔 휴리스틱 알고리듬을 제안하고, 제 4장에서는 제안된 알고리듬의 성능을 검증하기 위해 모의실험에 사용된 네트워크 모형과 최적해 및 Conde[10]의 휴리스틱 알고리듬과 비교한 결과를 제시하며, 그리고 결론이 마지막 제 5장에 이어진다.

2. 문제 정의

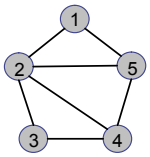
본 연구에서는 선후 제약 조건을 만족하면서 간격으로 주어진 제품분해 수익을 최대화하기 위한 분해의 깊이와 이에 따르는 분해순서를 결정하는 제품분해순서 문제를 다룬다. 선후 제약조건이란 제품 분해과정에 특정 분해공정이 먼저 수행되어야 함을 의미하며, 분해작업 시간은 순서 의존적이다.

순서 의존적인 작업시간과 작업선후관계를 고려하면 분해순서문제는 modified directed graph of assembly state라는 DAG상에 표현 가능하다[16]. <그림 1(d)>는 Kang [16]이 제안한 modified directed graph of assembly state의 예를 나타낸다. <그림 1(a)>와 <그림 1(b)>는 볼베어링 부품과 그 조립구조를 간단히 표현한 연결그래프(liaison graph, [6])를 각각 나타내는데, 연결그래프의 각 마디는 부품을 가지는 부품간의 체결 여부를 표시한다. 그러므로 연결그래프의 가지의 제거는 해당 부품간의 분해를 의미하고, 제품의 분해과정은 연결그래프의 cut-set(연결그래프를 2개의 분리된 서브그래프로 분리하는 과정)으로 정의될 수 있으므로, 조립부품의 가능한 분해순서는 순서 의존적인 작업시간을 고려하지 않으면 <그림 1(c)>와 같이 표현된다. <그림 1(c)>에서 마디는 분해단계에서 제품의 분해 상태를 표현하는데, 예를 들어, 좌측 마디는 모든 부품들이 체결된 상태를, 우측 마디는 모두 분해된 상태를 표시하고, 가지는 분해 작업을 나타내므로, 분해순서계획 문제는 이 DAG상에서 최장경로문제가 된다. 순서 의존적인 작업시간을 고려할 경우, 각 마디(node)의 수를 들어오는 가지(input arc)의 수만큼 반복하여 <그림 1(d)>와 같이 표현된다.

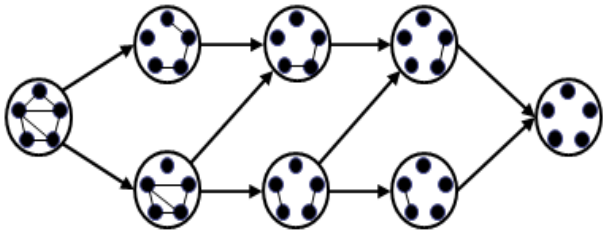


[Source: Penev and de Ron, 1996]

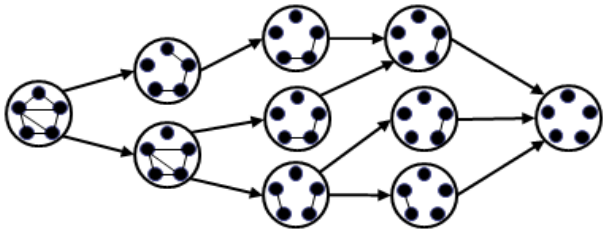
(a) A Ball Bearing assembly



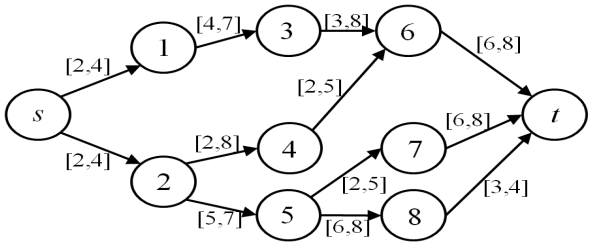
(b) The Liaison graph



(c) A Directed Graph of Assembly state w/o sequence dependent operation times



(d) A Modified Directed Graph of Assembly state with sequence dependent operation times



(e) Example of an interval graph based on the modified digraph of assembly states

<그림 1> Graphical Disassembly Representation

2.1 DAG 상에서 최대후회 최소화 모형

임의의 DAG($\mathbf{G} = (\mathbf{N}, \boldsymbol{A})$, $|\mathbf{N}| = n$, $|\boldsymbol{A}| = m$)상에서 가지의 가중치가 다음과 같이 주어진 경우를 가정하자. 그래프 상의 임의의 가지(i, j)의 가중치 c_{ij}는 그 값의 간격($c_{ij} \in [l_{ij}, u_{ij}]$)만이 알려져 있다. 임의의 c_{ij}는 구간 $[l_{ij}, u_{ij}]$에서 어느 값이던 취할 수 있으며, 이 값은 다른 가지의 가중치와는 독립적이다. 본 연구에서는 가지의 가중치가 간격으로 주어진 DAG를 편의상 *interval graph*라고 부르겠다. <그림 1(e)>는 임의의 interval weight를 <그림 1(d)>의 가지에 각각 부여한 것이다. 시작마디 s에서 목적마디 t까지 가능한 경로는 4가지이며, 이 경로들은 모두 특정한 상황에서 최적경로가 될 가능성이 있다.

Interval graph상에서의 최장경로문제의 0-1 정수계획법 모델은 아래 문제 **(P)**로 표현된다.

$$(\mathrm{P}) \quad \max \sum_{ij \in A} c_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{0j \in A} x_{0j} = 1 \tag{2}$$

$$\sum_{ij \in A} x_{ij} \geq \sum_{jk \in A} x_{jk} \quad \text{for all } j \in N \tag{3}$$

$$x_{ij} \in 0,\ 1 \quad \text{for all} \quad (i,\ j) \in A \tag{4}$$

$$c_{ij} \in [l_{ij},\ u_{ij}], \quad \text{for all} \quad (i,\ j) \in A \tag{5}$$

목적함수(1)의 계수 c_{ij}는 분해 작업의 비용과 분해의 결과로써 생성되는 부품 또는 서브어셈블리에서 발생하는 수익의 차이다. 이진 결정변수 x_{ij}는 분해 작업 (i, j)가 수행되면 1, 아니면 0이다. 식 (2)와 식 (3)은 일반적은 네트워크 흐름문제의 선형계획법 모형에서 사용되는 node balance equation들이고, 식 (3)에서 부등식은 불완전 분해공정일 때 성립한다. 여기서 목적함수(1)의 계수 c_{ij}는 간격으로 주어져 일반적인 선형계획법으로 풀이할 수 없으므로, 원문제의 목적함수에 최대후회 최소화 기준(minimax regret criterion)을 적용할 때, 이 문제를 최대후회최소화 최장경로문제(minmax regret longest path problem)이라 하고. 다음과 같이 정의된다[2, 3].

Interval graph에서 최장경로문제의 모든 가능해의 집합을 Ω라 하자. 그러면, Ω는 시작 노드로부터 최종 노드에 이르는 모든 가능한 경로들의 집합에 해당한다. 그러면, 최장경로문제는 다음과 같이 재정의 된다.

$$\max_{\mathbf{x}\in\Omega} \mathbf{cx} \tag{6}$$

where $\mathbf{c}\in\{\mathbf{c} = (c_{01}, \cdots, c_{n\text{-}1\ n}) \mid l_{ij} \leqq c_{ij} \leqq u_{ij},\ \forall(i, j) \in\mathbf{A}\}$ and $\mathbf{x} \in \Omega=\{x = (x_{01}, \cdots, x_{n\text{-}1n}) \mid x_{ij} \in (0, 1),\ \forall(i, j) \in \mathbf{A}\}$.

임의의 $\mathbf{x}, \mathbf{y} \in \Omega$에 대하여, $\mathbf{x}$와 $\mathbf{y}$간의 최대편차(=최대후회)는 다음과 같이 정의된다.

$$R(\mathbf{x},\mathbf{y}) = \max_{\mathbf{c}\in\Delta}(\mathbf{cy} - \mathbf{cx}) \tag{7}$$

where $\Delta = \{\mathbf{c} = (c_{01}, \cdots, c_{n\text{-}1n}) \mid c_{ij} = l_{ij}$ or $c_{ij} = u_{ij},\ \forall(i, j)\in\mathbf{A}\}$.

식 (7)에서 $\mathbf{x}$의 후회를 최대화하기 위한 목적함수의 계수 c_{ij}는 다음과 같이 결정된다[14].

$$c_{ij} = \begin{cases} l_{ij} & if\ arc_{ij} \in \mathbf{x} \\ u_{ij} & otherwise. \end{cases} \tag{8}$$

식 (8)은 가지 (i, j)가 경로 x에 포함되면, l_{ij}값을 그 외의 경우는 u_{ij}값을 취하는 경우 식 (7)의 우변이 최대가 되고, 이와 같은 특성을 *c-consistency*라 한다[14]. 이는 주어진 경로 $\mathbf{x}$의 최대후회를 구하기 위해서 모든 가능한 시나리오 중에서 오직 목적함수 계수 c_{ij}의 양극단의 값(l_{ij}, u_{ij})만을 고려하여도 됨을 의미하며, 모든 가능한 시나리오의 수는 $|\mathbf{S}| = 2^m\ (|A| = m)$이고, 식 (8)을 이용하여 임의의 주어진 $\mathbf{x}\in\Omega$의 최대후회는 다음과 같이 정의된다.

$$R_{\max}(\mathbf{x}) = \max_{\mathbf{y}\in\Omega} R(\mathbf{x},\mathbf{y}) \tag{9}$$

$$= \max_{\mathbf{y}\in\Omega}\ \max_{\mathbf{c}\in\Delta}(\mathbf{cy} - \mathbf{cx}) \tag{10}$$

식 (9), 식 (10)의 해($\mathbf{y}$)는 주어진 x의 최악대응경로(worst-case alternative path 또는 최악대응해 worst-case alternative solution)라 한다. 식 (10)을 이용하여 식 (6)을 최대후회 최소화 문제로 변형하면 다음과 같다.

$$\min_{\mathbf{x}\in\Omega} R_{\max}(\mathbf{x}) = \min_{\mathbf{x}\in\Omega}\ \max_{\substack{\mathbf{y}\in\Omega \\ \mathbf{c}\in\Delta}}(\mathbf{cy} - \mathbf{cx}) \tag{11}$$

그러므로 식 (11)의 해는 모든 가능경로 중에서 최대후회가 가장 작은 경로이며, a minmax regret longest path 또는 a *robust deviation longest path*라 한다.

<표 1>은 <그림 1(e)>의 예제에서 가능한 모든 경로들의 최대후회를 나타낸다. 경로(s - 1 - 3 - 6 - t)의 최대후회는 다음과 같이 구해진다. 경로(s - 1 - 3 - 6 - t)를 선택했을 때, 만약 경로(s - 2 - 4 - 6 - t)가 최상의 선택이었다면, 그 후회는 $(4 + 8 + 5 + 6) - (2 + 4 + 3 + 6) = 8$이 된다. 여기서, 경로($s$ - 2 - 4 - 6 - t) 상의 가지 (6 - t)의 값은 가지(6 - t)가 경로(s - 1 - 3 - 6 - t)와 공유되므로, c-consistency에 의해 6으로 결정된다. 이렇게 경로(s - 1 - 3 - 6 - t)의 후회는 다른 3개의 경로와 비교하여 그 중 가장 큰 값(경로 s - 2 - 5 - 7 - t와의 편차 9)이다. <표 1>에 따르면, 최소 최대후회 경로는 경로(s - 1 - 3 - 6 - t)이며 이 때 최대후회는 9임을 알 수 있다.

2.2 Karaşan's MILP Formulation

제 2.1절에서 정의한 최대후회 최소화 최장경로문제의 수리적 해법으로 Karaşan et al.[21]이 제안한 MRSP 문제의 MILP 모형을 최장경로문제로 바꾸어 표시하면 다음과 같다.

$$\text{(P1) Min} \quad y_n - \sum_{(i,j)\in A} l_{ij}x_{ij} \tag{12}$$

<표 1> Maximum regret of each path in the example <그림 1(e)>

Path	*Length*	*worst-case alt. path*	*Length*	*Max Regret*
s-1-3-6-t	15	*s-2-5-7-t*	24 = 4+7+5+8	9
s-2-4-6-t	12	*s-1-3-6-t*	25 = 4+7+8+6	13
s-2-5-7-t	15	*s-1-3-6-t*	27 = 4+7+8+8	12
s-2-5-8-t	16	*s-1-3-6-t*	27 = 4+7+8+8	11

subject to

$$y_i + u_{ij} - (u_{ij} - l_{ij})x_{ij} \le y_j \quad \forall (i,j) \in A \tag{13}$$

$$-\sum_{i \in \Gamma^-(j)} x_{ij} + \sum_{k \in \Gamma^+(j)} x_{jk} \le \begin{cases} 1 & j = 0 \\ 0 & j = 2,\ldots,n-1 \\ -1 & j = n \end{cases} \tag{14}$$

$$y_0 = 0$$
$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in A$$
$$y_j \ge 0 \quad j = 1,2,\ldots,n$$

여기서 결정변수 x_{ij}는 가지(i, j)가 최장경로에 속하면 1, 아니면 0의 값을 가지며, 결정변수 y_{ij}는 시작노드 0로부터 노드 i까지의 최장경로의 거리를 나타낸다. 식 (12)는 식 (11)의 MILP 모형으로 경로 $\mathbf{x}$의 최대후회를 나타내는데, y_n은 각 가지가 u_{ij}값을 가지는 최장경로로써 식 (11)의 우변항 중 $\mathbf{cy}$에 대응하고, 식 (12)의 두 번째 항은 식 (11)의 우변 $\mathbf{cx}$에 대응한다. 제약식 (13)는 Karaşan 모형의 핵심으로 x_{ij} 변수의 값에 따라 수식 (8)의 c-consistency에 의해 유도되는 시나리오 하에서, 노드 0부터 i까지 최장경로의 길이(결정변수 y_i)를 구한다. 즉, 가지(i, j)가 경로 $\mathbf{x}$에 포함되면, $y_j \ge y_i + l_{ij}$이고, 그렇지 않으면 $y_j \ge y_i + u_{ij}$가 된다. 나머지 제약들(12)는 일반적인 경로탐색문제의 모형과 다르지 않다. 단, $\Gamma^-(j)$와 $\Gamma^+(j)$는 각각 노드 j로 들어오는 가지들의 집합과 노드 j로부터 나가는 가지들의 집합을 나타낸다.

주어진 MRLP문제는 *NP*-hard로 문제의 규모가 커질 경우 실시간으로 최적해를 구할 수 없으므로[4, 34], 현실적인 접근법으로써 탐색해 기법이 필요하다. 다음 절에서는 동적계획법 기반의 효과적인 heuristic algorithm을 제안한다.

3. A Heuristic Algorithm

Kang[18]은 제품 분해계획을 수행하기 위한 최대후회 최소화 최장경로문제의 최적해를 구하는 동적계획법(Dynamic Programming, DP) 모형을 개발하였는데, 그 DP 모형은 다음과 같다.

Notation

$\mathbf{c}^u$, $\mathbf{c}^l$: (scenario u and l) : 모든 가지의 가중치가 upper bound 또는 lower bound로 설정될 때, 가중치 벡터

$\mathbf{c}^s(\mathrm{x})$: (scenario induced by path $\mathbf{x}$) : 임의의 경로 $\mathbf{x}$가 주어진 경우 c-consistency에 의해 결정되는 가지의 가중치. 즉, 경로 $\mathbf{x}$상의 모든 가지의 가중치는 lower bound로, 그렇지 않은 가지의 가중치는 upper bound 값,

Φ : 최악대응경로에 제외되어야 하는 가지의 집합

$\mathbf{x}^j$: 노드 0에서 노드 j까지의 최대후회최소화 최장경로

$\mathbf{y}(\mathbf{x})$: 임의의 주어진 경로 x의 최악대응경로.

$R_j(\Phi)$: 주어진 Φ를 고려한 시작마디부터 마디 j까지 최소-최대후회 경로($\mathbf{x}^j$)의 최대후회

Recursive equation of Kang's DP model

$$R_j(\Phi) = \underset{i \in \mathbf{V}(j)}{Min} \{Max[R_i(\Phi) - u_{ij}, R_i(\Phi \cup arc(i,j)) - l_{ij}]\} \tag{15}$$

위 Kang[18]의 recursive equation의 핵심은 다음과 같이 3가지로 요약된다.

1) 만약, 네트워크상의 모든 가지의 길이가 non-negative라면, 후회를 최대화하기 위해, 경로 $\mathbf{x}$가 시작노드 0부터 네트워크상의 임의의 노드 i까지의 경로라도 $\mathbf{y}(\mathbf{x})$는 항상 시작노드 0에서 목적노드 n까지의 최장경로이다. 특히, $\mathbf{x}^0$(= NULL)의 최악대응경로 $\mathbf{y}(\mathbf{x}^0)$는 c^u(scenario u)하에서 시작노드 0에서 목적노드 n까지의 최장경로이고, 이 때 $\mathbf{x}^0$의 최대후회는 $\mathbf{y}(\mathbf{x}^0)$의 시나리오 s^u하에서의 길이다.
2) 임의의 노드 i까지의 최대후회최소화 최장경로 $\mathbf{x}^i$와 그 최악대응경로 $\mathbf{y}(\mathbf{x}^i)$를 알 때, 가지(i, j)가 $\mathbf{y}(\mathbf{x}^i)$에 포함되지 않으면 경로 $\mathbf{x}^{ij}$(= $\mathbf{x}^i$∪가지(i, j))의 최악대응경로 $\mathbf{y}(\mathbf{x}^{ij})$는 $\mathbf{y}(\mathbf{x}^j)$와 같고, 그 최대후회는 $R_{ij} = R_j - l_{ij}$이다.
3) 가지(i, j)가 $\mathbf{y}(\mathbf{x}^j)$에 포함되면, 두 가지 경우로 나누어 최대후회를 계산한다.
   3-1) $\mathbf{y}(\mathbf{x}^{ij})$에 가지$(i, j)$가 포함된다고 가정하면, 이때 x^{ij}(= $\mathbf{x}^j$∪가지(i, j))의 최대후회는 $R_{ij} = R_j - u_{ij}$ 이고, 식 (15)의 우변 첫 항목 $R_i(\Phi) - u_{ij}$ 로 표현되며,
   3-2) $\mathbf{y}(\mathbf{x}^{ij})$에 가지$(i, j)$가 포함되지 않는다고 가정하면, 이때 x^{ij}를 구하기 위해서 노드 i로 재귀하는데, 이때 가지(i, j)가 x^j의 최악대응경로 $\mathrm{y}(\mathbf{x}^j)$에 포함되지 못하도록 제한한 임시 네트워크 G'(= G\가지(i, j))상에서 최대후회최소화 최장경로의 부분경로 $\mathrm{x}^{j'}$와 그 최악대응경로 $\mathrm{y}(\mathbf{x}^{j'})$를 찾는다. 이 식은 식 (15)의 우변 두 번째 항목 $R_i(\Phi \cup arc(i,j)) - l_{ij}$ 로 표현되어 있다.

주어진 문제가 전형적인 *NP*-hard인 것을 고려하면, 위의 동적계획법 모형은 대형 문제를 해결하는데 적합하지 않다[5]. 그러므로 이 장에서는 대형 문제를 효과적으로 수용 가능한 계산시간 내에 풀기 위하여, 동적계획법 모형에 기반을 둔 greedy heuristic 알고리듬을 제안한다.

3.1 Description of the heuristic algorithm

본 연구에서 제안하는 알고리듬의 기본 구조는 Dijkstra's algorithm과 유사하다. Dijkstra's algorithm에서는 시작노드 0에서 임의의 노드 i까지의 최단경로($d[i]$)를 알고, 가지(i, j)가 존재할 때, 시작노드 0에서 노드 j까지의 최단경로는 i까지의 최단경로에 가지(i, j)를 연장함으로써 얻을 수 있고, 이 경로의 평가척도는 경로 길이로 $d[i] + w(i, j)$가 되며, 이 거리가 현재의 $d[j]$보다 작으면 $d[j]$를 업데이트 한다[33]. 본 연구에서 제안하는 알고리듬에서는 평가척도가 거리가 아닌 최대후회가 되고, 경로 $\mathbf{x}^{ij}(=\mathbf{x}^i \cup$가지 $(i, j))$의 최악대응경로는 식 (8)에 의해 각 가지의 가중치가 시나리오 $\mathbf{c}^s(\mathbf{x}^{ij})$로 주어진 DAG 상에서 최장경로문제의 해이며, DAG 상에서 Dijkstra's algorithm을 이용하여 구할 수 있다[19].

3.2 Reduction Rule

각 가지를 부분경로에 추가할 때마다 maximum regret을 구하기 위해 매번 Dijkstra's algorithm을 수행하는 것은 계산시간을 증가시키므로, Kang의 recursive equation에 대한 설명 2)에서 언급한 바와 같이, 임의의 노드 i까지의 최대후회최소화 최장경로 $\mathbf{x}^i$와 그 최악대응경로 $\mathbf{y}(\mathbf{x}^i)$를 알 때, 가지(i, j)가 $\mathbf{y}(\mathbf{x}^i)$에 포함되지 않으면 경로 $\mathbf{x}^{ij}(=\mathbf{x}^i \cup$가지$(i, j))$의 최악대응경로 $\mathbf{y}(\mathbf{x}^{ij})$는 $\mathbf{y}(\mathbf{x}^i)$와 같고, 그 최대후회는 $r_{ij}=r_j-l_{ij}$로 간단히 계산할 수 있다.

3.3 Pseudo-code

<그림 2>는 이 알고리듬의 pseudo-code를 나타내는데, 각 행의 의미는 다음과 같다.

- 0~1행 : 초기화
- 2행 : 시작노드까지의 최적경로($\mathbf{x}^0$ = NULL)의 최악대응경로($\mathbf{y}^0$)를 구한다. $\mathbf{y}^0$는 NULL path의 최악대응경로이므로, c-consistency에 의해 모든 가지들의 가중치가 u_{ij}로 결정된 시나리오 u 하에서 최장경로에 해당하고, 이를 찾기 위한 과정($\mathrm{Max}_{\mathbf{y}}[\mathbf{c}''\mathbf{y}]$)은 DAG상에서 Dijkstra's algorithm을 변형하여 최장경로를 구한다. $\mathbf{x}^0$의 길이가 0이므로 $\mathbf{x}^0$의 최대후회 R^0는 $\mathbf{y}^0$의 길이와 같다.
- 3행 : 그래프 상의 마디들을 순회하며 4~13행을 반복 수행한다.
- 4행 : 반복문으로써 5~13행을 반복하게 되는데, 각 마디별로 해당 마디에 들어오는 가지들 중에서 최대후회가 가장 작은 가지를 선택하게 된다.
- 5행 : 가지(i, j)를 포함하는 최적의 경로는 마디 i까지의 최적경로($\mathbf{x}^i$)와 가지(i, j)의 합집합이다.
- 6~8행 : 만약 해당 가지(i, j)가 마디 i까지의 최적경로($\mathbf{x}^i$)의 최악대응경로 $\mathbf{y}^i$에 포함되지 않으면, 3.2절에 설명된 reduction rule에 따라, 주어진 경로($\mathbf{x}^{ij}$)의 최악대응경로 및 최대후회를 간단히 계산한다.
- 9~11행 : 만약 해당 가지(i, j)가 마디 i까지의 최적경로($\mathbf{x}^i$)의 최악대응경로 $\mathbf{y}^i$에 포함되면, 주어진 경

```
    Procedure: Algorithm MinMaxRegret(G = (N, A), c_ij∈[l_ij, u_ij]) for each arc (i, j)∈A, source)
0   for each node j∈ N                                          // Initializations
1       R_j: = infinity                                        // all maximum regret are set to infinity
2   x^0 = NULL, R_0 = Max_y[c''y], let y^0 be the optimal solution of Max_y[c''y]   // current node = source node
3   for each node j∈N                                          // The main loop
4       for each i∈ N such that (i, j)∈A, let                  // for each input arc of node j
5           x^ij : = x^i ∪ (i, j)                              // extends path to arc(i, j)
6           if arc(i, j)∉y^i                                   // if arc(i, j) belongs to y^i
7               R_ij : = R_j − l_ij        // the maximum regret of path x^ij is calculated
8               y^ij : = y^i               // the worst-case alternative of x^ij is the worst-case alternative of x^i
9           else
10              R_ij : = Max_y[c^s(x^ij)y] − c^l x^ij
                                   // solve the longest path problem under the scenario induced by x^ij
11              y^ij : = the optimal solution of Max_y[c^s(x^ij)y]   // the worst-case alternative of x^ij
12          if R_ij < R_j              // take the least maximum regret among all R_ij
13              R_j : = R_ij, x^j : =x^ij, y^j : =y^ij          // update minmax regret path until node j
```

<그림 2> Pseudo-code of the proposed algorithm

로($\mathbf{x}^{ij}$)의 최악대응경로 및 최대후회를 재계산한다. $\mathrm{Max}_{\mathbf{y}}[\mathbf{c}^{s}(\mathbf{x}^{ij})\mathbf{y}]$는 주어진 경로($\mathbf{x}^{ij}$)에 의해 결정되는 시나리오($\mathbf{c}^{s}$) 하에서 최장경로를 구하는 과정을 나타내며, Dijkstra's algorithm을 이용하여 최장경로를 구한다. 시나리오($\mathbf{c}^{s}$)는 c-consistency에 의해 경로($\mathbf{x}^{ij}$) 상의 모든 가지의 가중치는 l_{ij}를 나머지는 u_{ij}값을 갖는 경우를 말한다.

- 12~13행 : 9~11행에서 구해진 최대후회가 기존의 최대후회보다 작으면 최소최대후회경로(($\mathbf{x}^{j}$))를 현재의 경로($\mathbf{x}^{ij}$)로 갱신한다. 즉, 들어오는 가지들 중 최대후회가 가장 작은 가지를 포함한 경로를 마디 j까지의 최적 경로로 결정한다.

3.4 Computational complexity

Dijkstra's algorithm의 실행시간을 K라 하자. 그러면, 주어진 네트워크가 $\mathbf{G} = (\mathbf{N}, A)$, $|\mathbf{N}| = n$, $|A| = m$라 할 때, 대문자 O표기법으로 이 알고리듬의 실행시간을 나타내면, 최악의 경우 $O(mK)$이다. 즉, 만약 7행의 최장경로 문제를 선형탐색 기법에 기반을 둔 Dijkstra's algorithm으로 구현했을 때, Dijkstra's algorithm의 실행시간은 $O(n^2)$이나, c-consistency에 따라 각 가지의 가중치를 설정하는 과정이 $O(m)$시간만큼 필요하므로, 위 알고리듬의 실행시간은 $O(m^2n^2)$이 되는데, 실제 문제에서는 6행의 조건, 즉 가지(i, j)가 $\mathbf{y}(\mathbf{x}^{i})$에 포함되는 경우가 많지 않으므로, 평균적인 실행시간은 훨씬 짧다.

4. 모의실험에 의한 검증

분해순서계획을 표현하는 modified directed graph of assembly state를 모사하여 랜덤하게 생성된 interval graph 상에서 수행된 일련의 실험 결과를 요약하여 설명한다. 이 실험의 목적은 본 연구에서 제시한 알고리듬의 성능과 효율을 살펴보기 위하여 Karaşan의 MILP 모형에 의한 최적해와 Conde[10]가 제시한 알고리듬 및 본 연구에서 제안한 알고리듬의 탐색해 간의 편차의 비(GAP (%) = (탐색해－최적해)/최적해 × 100)를 비교하고, 네트워크 모형에 따른 문제의 난이도 및 알고리듬의 성능을 분석한다. 또한, 일반적으로 modified directed graph of assembly state로 모형화 된 분해순서계획 문제가 대형 최대후회 최소화 최장경로문제로 변형되므로, 이런 대형 최대후회 최소화 최장경로문제를 풀기 위한 계산수행시간의 증가 경향을 살펴본다.

<표 2> Summary of computational experiments

layers	width	c	d	Comp. Time						GAP				Correct %		Proposed algorithm #.shortest path calls
				MIP		Conde's heuristic		Proposed algorithm		Conde's heuristic		Proposed algorithm		Conde's algorithm	Proposed algorithm	
				mean	STD	mean	STD	mean	STD	mean	STD	mean	STD			
27	25	10	0.3	43.37	12.356	0.004	0.007	0.038	0.007	3.77%	0.043	0.54%	0.014	29%	74%	18.2
			0.9	41.35	10.570	0.004	0.007	0.038	0.007	1.99%	0.027	0.13%	0.006	46%	93%	19.9
		20	0.3	44.97	13.410	0.004	0.007	0.037	0.007	2.65%	0.036	0.37%	0.010	37%	80%	17.9
			0.9	41.68	11.295	0.004	0.007	0.038	0.007	2.08%	0.034	0.24%	0.010	42%	87%	18.7
	50	10	0.3	716.94	122.970	0.031	0.003	0.267	0.026	1.39%	0.024	0.12%	0.005	50%	92%	18.6
			0.9	731.45	127.971	0.031	0.002	0.283	0.033	0.86%	0.011	0.15%	0.007	61%	89%	20.2
		20	0.3	772.43	121.834	0.030	0.004	0.260	0.032	1.99%	0.028	0.15%	0.007	44%	89%	18.7
			0.9	792.69	159.293	0.029	0.006	0.267	0.031	1.19%	0.025	0.07%	0.003	59%	93%	19.3
52	25	10	0.3	415.69	1267.64	0.012	0.007	0.177	0.014	2.14%	0.030	0.24%	0.005	21%	74%	33.2
			0.9	308.08	135.94	0.012	0.007	0.200	0.020	1.82%	0.017	0.19%	0.004	21%	72%	36.0
		20	0.3	363.76	188.36	0.012	0.006	0.203	0.055	2.30%	0.022	0.35%	0.008	20%	67%	32.5
			0.9	452.82	1176.57	0.011	0.007	0.216	0.025	2.04%	0.017	0.35%	0.008	19%	68%	36.2
	50	10	0.3	4205.96	830.75	0.060	0.006	0.902	0.088	1.17%	0.015	0.06%	0.002	50%	94%	35.0
			0.9	5277.18	9511.45	0.061	0.005	0.966	0.087	1.21%	0.012	0.12%	0.003	52%	86%	36.7
		20	0.3	5515.10	1306.77	0.066	0.007	1.009	0.092	1.42%	0.018	0.22%	0.005	50%	85%	34.0
			0.9	4455.14	1024.27	0.060	0.005	0.915	0.076	1.36%	0.017	0.10%	0.003	50%	90%	35.7

주) GAP(%) = (탐색해－최적해)/최적해 × 100(%).

4.1 네트워크 모형 및 가중치 생성

<그림 1(d)>에서 처럼 modified directed graph of assembly state는 비순환 계층형 그래프(*acyclic layered* graph)이다. 계층형이란 그래프의 노드들이 공통원소를 갖지 않는 노드들의 부분집합으로 분리될 수 있으며, 이 때 각 부분집합의 cardinality는 주어진 폭(width, w)에 해당한다. <그림 1(d)>는 최대 폭이 3인 5 계층 그래프의 예이다. Modified directed graph of assembly state에서 폭은 각 분해단계에서 가능한 분해의 종류를 나타내며, 계층의 수는 부품 수에 나타낸다. 각 층에 해당하는 모든 노드들이 다음 층에 해당하는 모든 모드들과 직접 연결된 경우 완전(*complete*) 연결이라 하는데, modified directed graph of assembly state는 일반적으로 희소(sparse) 연결이지만, 본 모의실험에서는 완전연결 비순환 계층형 그래프를 기반으로 계층형 interval graph를 생성하였다. 실제 분해순서계획에서 연결의 수가 많음은 분해 가능한 경우의 수가 많음을 의미하고, 분해순서계획을 더욱 어렵게 한다.

각 가지의 간격 가중치 $c_{ij} \in [l_{ij}, u_{ij}]$는 다음과 같이 랜덤하게 생성된다[21]. 균등분포(uniform distribution)를 따르는 난수 $c_{ij} \in [1, c]$를 생성하고, $l_{ij} \in [(1-d)c_{ij}, (1+d)c_{ij}]$와 $u_{ij} \in [l_{ij}, (1+d)c_{ij}]$를 균등분포에 준하여 랜덤하게 생성한다. 여기서, c와 d는 네트워크의 불확실성의 정도를 조절하는 모수로, c가 클수록 각 가지별 가중치의 중앙값 간의 차이가 크게 되고, d가 클수록 가중치의 간격이 커진다.

모든 테스트는 C++언어로 코딩되고 MS Visual C++ ver.6에 의해 컴파일 되었으며, Intel Core2Duo @ 2.33GHz CPU/2GB main memory를 가진 컴퓨터에서 수행되었다. MILP 모형의 풀이는 오픈소스 선형계획법 라이브러리인 Cbc(Coin-or branch and cut, https://projects.coin-or.org/Cbc) ver. 1.1.2에 의해 수행되었는데, Cbc는 대형 LP, MILP(Mixed Integer Linear Programming) 및 관련 문제를 풀기 위해 개발된 오픈소스 소프트웨어로써, 대형 벤치마킹 문제들의 비교결과 상용 소프트웨어인 ILOG CPLEX 12.0에 비해서 5배 정도 느린 것으로 알려져 있으나[25], 최적해와 휴리스틱 알고리듬에 의한 탐색해 사이의 편차의 비(GAP)를 알기 위해 사용되므로 수행속도는 직접적인 비교 대상이 아니며, 단지 최적해를 구하는데 필요한 시간의 경향을 나타내기 위해 사용되었다. 실질적으로 대량의 실험에서 다수의 컴퓨터를 이용하기 위해서, Free-code MILP solver 중 최상의 성능을 보이는 것 중의 하나인 Cbc가 사용되었다.

4.2 실험 결과

본 연구에서 제시된 알고리듬을 검증하기 위하여, layered interval graph의 크기는 부품수 = {26개, 51개}(L = {27, 52}와 그래프의 폭 w = {25, 50}을 각각 2가지 경우로 나누어 4가지 크기의 그래프를 생성하였는데, 각 그래프의 cardinality는 $|\mathbf{N}| = n = (L-2)w + 2$, $|\boldsymbol{A}| = m = 2*w + (n-3)w^2$으로 정의된다. 불확실성의 정도를 결정하는 모수 c와 d에 대하여 각각 c = {10, 20}, d = {0.3, 0.9}으로 하는 4가지 조건으로 모의 실험하였으며, 각 실험 조건 별로 100회씩 반복하였다.

<표 2>은 각 실험 결과를 요약하여 보여준다. <표 2>에서 첫 세열은 실험 조건(L, w, c, d)을 나타내며, MIP, Conde's heuristic algorithm과 본 연구에서 제안한 알고리듬(Proposed algorithm)의 수행도를 나타내는데, 각 열의 의미는 다음과 같다.

- Comp. Time : 소요된 연산시간 (seconds),
- GAP(%) = (탐색해 − 최적해)/최적해 × 100 (%),
- Correct % : 정답률, 100번의 반복 중 최적해를 찾은 비율,
- #. shortest path calls : Dijkstra's algorithm을 이용하여 최악대응경로를 찾는 평균 횟수,
- Mean : 평균, STD(= STandard Deviation) : 표준편차.

<표 2>의 결과 Conde의 알고리듬의 평균 GAP은 1.84%인데 비해, 본 논문에서 제안한 알고리듬의 평균 GAP은 0.21%로 최적해에 근접한 해를 찾는다. 또한, 최적해를 찾는 비율도 83.3%로 Conde의 알고리듬의 40.1%에 비해 훨씬 높음을 알 수 있다. 즉, 계산 시간 면에서 Conde 알고리듬과 비교하면 상대적으로 많은 시간이 소요되나, GAP과 정답률 양면에서 확연한 차이를 보인다.

본 논문에서 제안한 알고리듬과 Conde의 알고리듬간의 성능 차이의 원인은 다음과 같이 두 가지로 요약할 수

<표 3> summary of performance

		layers	
		27	52
width	25	0.321%	0.283%
	50	0.124%	0.124%

(a) GAP

		layers	
		27	52
width	25	83.5%	70.2%
	50	90.7%	88.7%

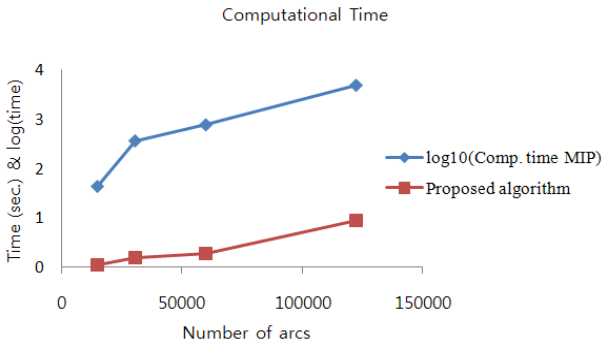
(b) Correct %

		c	
		10	20
d	0.3	0.242%	0.274%
	0.9	0.146%	0.190%

(c) GAP

		c	
		10	20
d	0.3	83.5%	80.2%
	0.9	85.0%	84.5%

(d) Correct %



<그림 3> Computational time of the proposed algorithm

있다. 첫째, Conde의 알고리듬에서는 주어진 경로의 최대후회를 계산할 때, 최악대응경로를 시작마디에서 목적마디까지의 경로로 보지 않고, 주어진 경로의 최종마디까지의 경로만을 고려함으로써 실질적 최악대응경로가 아니므로, 최대후회가 부정확한 문제가 있는데, 본 논문에서 제안한 알고리듬은 최적경로를 시작마디에서부터 각 마디로 확장해 가는 과정에 미리 잠재적 최악대응경로를 고려함으로써 이와 같은 문제를 해결한다.

둘째, 각 마디별로 잠재적 최적경로를 확장해 가는 과정에서, 주어진 경로의 최악대응경로를 찾는 것이 아니라, 기존의 최악대응경로 중 하나를 선택한다. 이는 본 논문에서 제안한 Reduction Rule과 기본 개념상으로 비슷하나, 본 연구에서 제안한 알고리듬의 경우 해당 대응경로가 주어진 경로의 최악대응경로임이 분명한 경우에만 기존의 최악대응경로를 이어받는데 반해, Conde의 알고리듬에서는 무조건 주어진 대응경로 중 가장 그 값이 큰 것을 선택함으로써 실제 최악대응경로를 찾지 못하는 경우가 많다. 본 논문에서 제안한 알고리듬에서는 Reduction Rule의 조건이 만족하지 않는 경우, 주어진 경로의 최악대응경로를 직접 구함으로써 해의 성능을 향상시키고 있다. 단, 이 경우 Dijkstra's algorithm을 이용하여 최장경로를 구하는 과정이 필요하여 수행시간이 증가하게 된다. 그러므로 Conde의 알고리듬에 비해서는 수행시간이 길지만, 그 수행시간의 증가를 고려해도 본 알고리듬의 수행시간은 충분히 짧다.

<그림 3>은 본 연구에서 제안한 알고리듬의 연산시간을 요약하여 보여준다. 가로축은 가지의 수(m)를, 세로축은 <표 2>에 표기된 연산시간(comp. time)을 나타내는데, MIP에 의한 최적해를 구하는 시간은 상대적으로 크므로 실제값 대신 상용로그값을 취해 log10(comp. time)으로 표기하였고, 본 연구에서 제시한 알고리듬의 경우 실제 연산시간(CPU time in seconds)을 표시하였다. <그림 3>의 연산시간의 증가추세를 보면, 비록 최악의 경우 알고리듬의 수행시간이 $O(m^2n^2)$을 따르나, Dijkstra's algorithm을 이용하여 최악대응경로를 찾는 평균 횟수가 많지 않으므로(<표 2>의 #. shortest path calls), 실질적으로는 연산시간이 m에 선형적으로 비례하여 증가함이 관찰된다. 참고로 본 논문에서 다루는 문제는 NP-Hard로써, 최적해를 구하는데 소요되는 시간은 문제의 크기가 증가함에 따라 기하급수적으로 증가한다. 예를 들어, 61개의 부품 수에 해당하는 layer × width = 62 × 30 크기의 문제를 28회 반복 실험한 결과, Karaşan et al.모형을 Cbc를 이용하여 최적해를 구하는데 평균 66787초(= 18시간 이상)이 소요되었으며, 그 중 최악의 경우에는 217625초(= 60시간 이상)의 수행시간이 소요되었다.

<표 3>는 제안된 알고리듬의 성능을 그래프의 모양(높이와 폭)과 불확실성의 정도를 결정하는 모수 c와 d에 따라 요약하여 나타낸다. <표 3(a)>와 <표 3(b)>는 그래프의 모양에 따른 GAP과 정답률(correct %)을 각각 나타내는데, 높이가 낮고(부품수가 적고, $L = 27$) 폭이 넓을 때(분해 가능한 방법이 많을 때, $w = 50$) 높은 성능을 보인다. 또한, <표 3(a)>와 <표 3(b)>는 불확실성의 정도를 결정하는 모수 c와 d에 따른 GAP과 정답률을 각각 나타내는데, $c = 10$, $d = 0.9$일 때 높은 정확도를 보임을 알 수 있다.

5. 결 론

본 연구에서는 순서 의존적인 분해 작업시간 및 회수된 소재의 가치가 간격으로 주어진 분해순서계획을 다루었다. 간격으로 주어진 목적함수의 계수를 다루기 위해, 불확실성 하에서 의사결정기준으로 사용되는 최대후회 최소화 기준을 적용하여 분해순서계획 문제를 가지의 가중치가 간격으로 주어진 DAG 상에서 최대후회 최소화 최장경로문제로 변환하고, 전형적인 *NP*-hard 문제인 대형 분해순서계획 문제를 효과적으로 풀기 위하여, 동적계획법에 기반을 둔 heuristic algorithm을 제안하였다.

제안된 알고리듬의 성능을 분석하기 위하여, 분해순서계획에서 사용되는 modified directed graph of assembly state와 비슷한 형태를 갖는 complete layered graph를 random하게 생성하여 Karaşan et al.이 제안한 MIP 모형과 Conde가 제안한 휴리스틱 알고리듬과 비교하였다. 모의시험 결과, Conde의 알고리듬과 비교하여 확연한 성능향상을 보였으며, 수행시간 측면에서도 충분히 빠르고, 특히, 이론적 수행시간인 $O(m^2n^2)$과 달리, 실제 실험에서는 $|A| = m$에 근사적으로 선형 비례하여 증가하는 것으로 관찰되었다.

가중치가 간격으로 주어진 DAG 상에서 최대후회 최

소화 최장경로문제는 본 연구에서는 분해순서계획을 위해서 사용되었으나, 일반적인 네트워크상에서 불확실성이 존재하는 최단경로문제 또는 PERT/CPM으로 잘 알려진 불확실성 하에서 최장경로문제 뿐만 아니라, 최단경로 및 최장경로 문제로 모형화 되는 불확실성 하에서의 의사결정 문제들에 적용 가능할 것으로 기대한다.

참고문헌

[1] 유호현; "일본 사례를 통해 본 도시광산의 미래", LG Business Insight : LG경제연구원, 1038 : 48-54, 2009.

[2] Aissi, H., Bazgan, C., and Vanderpooten, D.; "Min-max and min-max regret versions of combinatorial optimization problems : A survey," *European Journal of Operational Research*, 197(2) : 427-438, 2009.

[3] Assavapokee, T. and Realff, M. J.; "Min-Max Regret Robust Optimization Approach on Interval Data Uncertainty," *Journal of Optimization Theory and Applications*, 137(2) : 297-316, 2008.

[4] Averbakh, I. and Lebedev, V.; "Interval data minmax regret network optimization problems," *Discrete Applied Mathematics*, 138 : 289-301, 2004.

[5] Averbakh, I. and Lebedev, V.; "On the complexity of minmax regret linear programming," *European Journal of Operational Research*, 160(1) : 227-231, 2005.

[6] Bourjault, A.; "Contribution à une approche méthodologique de l'assemblage automatisé : elaboration automatique des séquences opératoires." Besançon : Université de Franche-Comté, Ph.D. thesis, 1984.

[7] Chanas, S. and Zieliński, P.; "The computational complexity of the criticality problems in a network with interval activity times," European Journal Of Operational Research, 136(3) : 541-550, 2002.

[8] Chanas, s. and Zieliński, P.; "On the hardness of evaluating criticality of activities in a planar network with duration intervals," Operations Research Letters, 31(1) : 53-59, 2003.

[9] Chevron, D., Binder, Z., Perret, R., and Horacek, P.; "Disassembling Process Modelling and Operations Planning under Imprecise Operation Time," 13th IFAC World Congress, san Francisco, 367-372, 1996.

[10] Conde, E.; "A minmax regret approach to the critical path method with task interval times," *European Journal of Operational Research*, 197(1) : 235-242, 2009.

[11] Eppstein, D.; "Finding the k shortest paths," SIAM Journal on Computing, 28(2) : 652-673, 1998.

[12] Erdös, G. and Kis, T.; "Disassembly sequence planning for products with defective parts in product recovery," *International Journal of Production Research*, 39 : 1203-1220, 2001.

[13] Homem De Mello, L. S. and Sanderson, A. C.; "A Correct and Compete Algorithm for the generation of Mechanical Assembly sequences," *IEEE Transactions on Robotics and Automation*, 7 : 228-240, 1991.

[14] Inuiguchi, M. and Sakawa, M.; "Minimax Regret Solution To Linear-Programming Problems with An Interval Objective Function," *European Journal Of Operational Research*, 86(3) : 526-536, 1995.

[15] Johnson, M. R. and Wang, M. H.; "Economical evaluation of disassembly operations for recycling, remanufacturing and reuse," *International Journal of Production Research*, 36(12) : 3227-3252, 1998.

[16] Kang, J.-G., Lee, D.-H., and Xirouchakis, P.; "Optimal Disassembly Sequencing with Sequence-Dependent Operation Times based on the Directed Graph of Assembly states," *Journal of the Korean Institute of Industrial Engineers*, 28(3) : 264-173, 2002.

[17] Kang, J.-G., Lee, D.-H., and Xirouchakis, P.; "Disassembly Sequencing with Imprecise Data : a Case Study," *International Journal of Industrial Engineering-Theory*, Applications and Practice, 10(4) : 407-412, 2003.

[18] Kang, J.-G.; "Robust Disassembly Sequencing with Interval Data," Proceedings of 6th Global Conference on Sustainable Product Development and Life Cycle Engineering, Busan, Korea, 109-113, 2008.

[19] Kang, J.-G.; "A heuristic algorithm to find the minimax regret longest path problem with interval lengths," *Journal of the Korean Management Engineers society*, 14(3) : 35-46, 2009.

[20] Kang, J.-G. and Xirouchakis, P.; "Disassembly sequencing for maintenance : a survey," Proceedings of the Institution of Mechanical Engineers, Part B : *Journal of Engineering Manufacture*, 220(10) : 1697-1716, 2006.

[21] Karaşan, O. E., Pinar, M. Ç., and Yaman, H.; "The Robust shortest Path Problem with Interval Data," available online at <http://www.ie.bilkent.edu.tr/~mustafap/pubs/>.

[22] Kim, H., Han, D., Jeong, H., and Lee, S.; "Design of RFID-based Integration System for Collection and Recycling Process of EOL Household Electric Appliances in Korea," *Journal of the society of Korea Industrial and Systems Engineering*, 32(2) : 120-131, 2009.

[23] Lambert, A. J. D.; “Optimal disassembly Sequence generation for combined material recycling and part reuse,” Proceedings of IEEE International Symposium on Assembly and Task Planning, 146-151, 1999.

[24] Lim, S.; “A Study on a Solution Approach to Fuzzy Linear Programs and Its Application to Fuzzy DEA Models,” *Journal of the Society of Korea Industrial and Systems Engineering*, 31(2) : 51-60, 2008.

[25] <http://plato.asu.edu/bench.html >.

[26] Montemanni, R. and Gambardella, L. M.; “An Exact Algorithm for the Robust Shortest Path Problem with Interval Data,” *Computers and Operations Research*, 31(10) : 1667-1680, 2004.

[27] Montemanni, R., Gambardella, L. M., and Donati, A. V.; “A branch and bound algorithm for the robust Shortest path problem with interval data,” *Operations Research Letters*, 32(3) : 225-232, 2004.

[28] Montemanni, R. and Gambardella, L. M.; “The robust Shortest path problem with interval data via Benders decomposition,” 4OR, 3(4) : 315-328, 2005.

[29] Park, H. S., Choi, H. W., Mok, H. S., Moon, K. S., and Sung, J. H.; “Method for Generating Optimal Disassembly Sequence of End-of-Life Car’s Parts,” *Journal of the Korean Society of Precision Engineering*, 20(9) : 188-196, 2003.

[30] Seo, K. K., Park, J. H., and Jang, D. S.; “Optimal disassembly Sequence using genetic algorithms considering economic and environmental aspects,” International Journal of Advanced Manufacturing Technology, 18(5) : 371-380, 2001.

[31] Tang, O., Grubbstrom, R. W., and Zanoni, S.; “Economic evaluation of disassembly processes in remanufacturing Systems,” *International Journal of Production Research*, 42(17) : 3603-3617, 2004.

[32] Tripathi, M., Agrawal, S., Pandey, M. K., Shankar, R., and Tiwari, M. K.; “Real world disassembly modeling and Sequencing problem : Optimization by Algorithm of Self-Guided Ants (AsGA),” Robotics and Computer-Integrated Manufacturing, 25(3) : 483-496, 2009.

[33] <http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm>.

[34] Zieliński, P.; “The computational complexity of the relative robust Shortest path problem with interval data,” *European Journal of Operational Research*, 158(3) : 570-576, 2004.