Journal of the Korean Institute of Industrial Engineers
Vol. 25, No. 1, pp. 21-34, March 1999.

조선 소조립 공정의 자동화를 위한 용접 로봇 스케줄링

[1]강장하 · [1]박성수 · [2]박경철 · [3]도기상

[1]한국과학기술원 산업공학과 / [2]한국통신 통신망연구소 / [3]삼성중공업 중앙연구소

Scheduling of Welding Robots for Shipyard Sub-assembly Process

Jang Kang · Kyungchul Park · Sungsoo Park · Kisang Do

We consider a scheduling problem arising in a shipyard subassembly welding process. There are four welding robots of gantry type, which perform the welding process for the subassemblies. Because the robots perform the welding operations at the same time, there is a possibility of collision between adjacent robots depending on the welding schedule. In this paper, we propose a heuristic method to find a welding schedule which does minimize the welding completion time while avoiding collision among the robots. The method consists of two phases: assignment and scheduling. In the assignment phase, we assign each welding line to a proper robot. In the scheduling phase, we determine the welding schedules for the robots so that collision is avoided. Computational experiences with the data which reflect the real situation are reported.

1. 서 론

로봇에 의한 조립 작업의 자동화는 작업의 균일화와 투입되는 노동력의 감소에 큰 도움이 된다. 이와 같은 이유로 전자 부품의 조립 공정이나 자동차의 조립 공정의 자동화는 오래 전부터 상당한 발전을 이루어 왔다. 그러나 선박 조립 공정에서의 자동화의 발전은 더딘 편이다. 이는 다음과 같은 선박 조립 공정의 특징에서 기인한다. 첫째, 작업 대상과 스케줄이 자주 변한다. 1대의 대형 선박을 건조할 때 각 부분에 들어가는 부분들은 각기 달리 설계되며 따라서 각 부분들은 다른 작업 스케줄에 따라 조립되고, 작업이 시작할 때마다 새로운 작업 스케줄을 생성해야 한다. 둘째, 조립 대상의 크기가 크고 조립(용접) 작업에 소요되는 시간이 길다. 따라서 전체 작업 시간을 감소시킬 다양한 방법들이 사용되고 이는 작업 스케줄을 어렵게 한다.

대형 선박의 조립 공정은 크게 소조립, 대조립, 탑재로 이루어진다. 소조립은 원하는 모양으로 잘린 철판들을 용접하여 선박의 작은 부분을 만드는 공정이고, 대조립은 소조립에서 만들어진 작은 부분들을 모아 큰 부분으로 만드는 공정이며, 탑재는 대조립에서 만들어진 선박의 부분들을 용접하여 최종적으로 배를 만드는 공정이다. 본 연구에서는 이들 중 자동화가 우선적으로 적용 가능한 소조립 공정의 자동화에 대해서 연구하였다.

소조립 공정에서는 바탕이 되는 넓은 철판과 이를 지탱하기 위해 사용되는 철판이 수직으로 용접된다[1]. 이렇게 하나의 철판과 이에 수직으로 놓이는 철판을 용접하는 것을 fillet 용접이라 한다. 이때 바탕이 되는 넓은 철판을 base panel이라 하고, base panel에 수직으로 용접되는 철판을 stiffener라 한다. Base panel 위에 stiffener를 붙이기 위해서 stiffener의 양쪽을 모두 용접해야 하기 때문에 두 개의 용접 로봇이 하나의 gantry 위에 설치되어 사용된다. 이러한 형태를 Macromini System이라 하며 [2], 한 stiffener의 양변을 동시에 용접할 수 있다. 따라서 stiffener의 양변을 따라 평행하게 나타나는 두 용접선의 중심에 새로운 용접선을 가정하고 이러한 새로운 용접선의 용접 스케

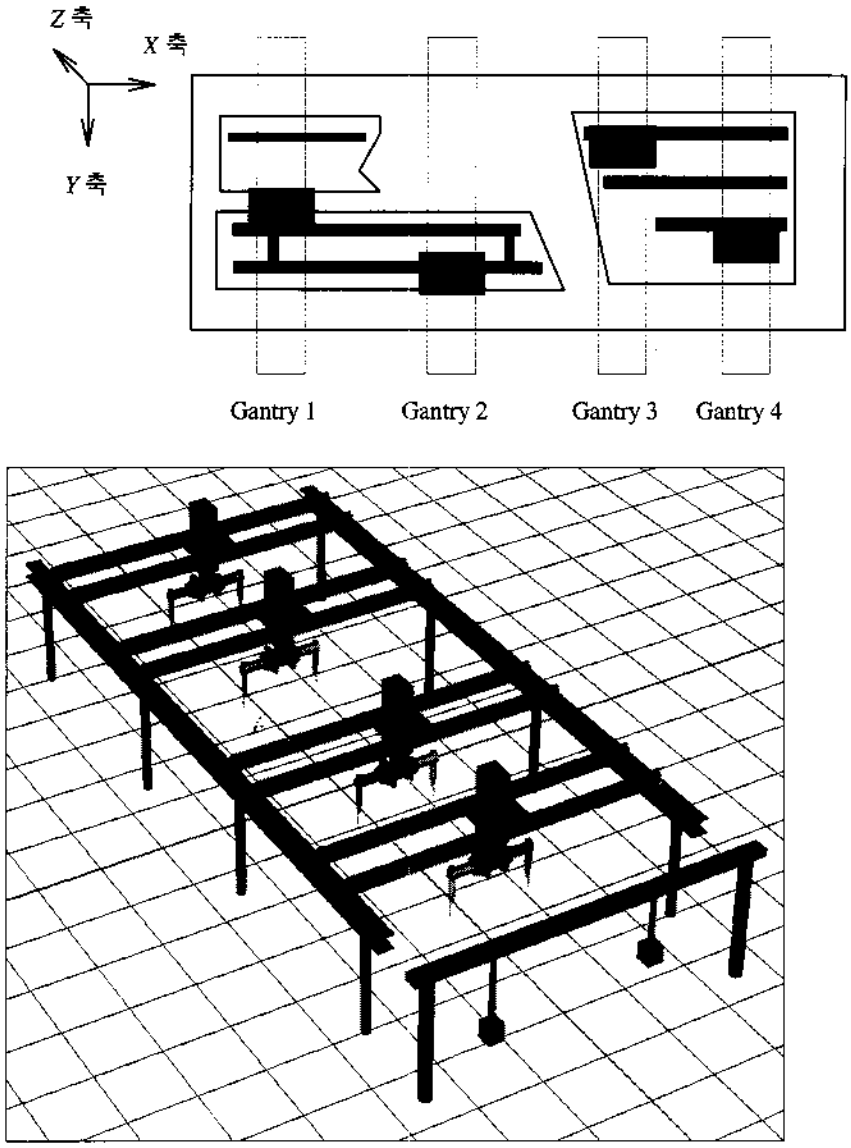

그림 1. 조선 소조립 자동화 용접 로봇 시스템.

줄을 구함으로써 원래 용접선에 대한 스케줄링을 구할 수 있다.

기존의 조선 소조립 공정에서는 주로 작업자가 소형의 반자동 로봇을 끌고 다니며 용접 작업을 수행한다. 또한 일부 회사에서 사용하고 있는 로봇 자동화 공정에서도 개별 로봇이 서로 간섭을 일으키지 않는 상황에 대해서만 자동화 작업이 수행되고 있다. 그런데 대체로 조선 소조립 공정의 작업 영역은 넓고, 용접 속도는 느리기 때문에, 한 대의 용접 로봇이 전체를 용접하는 것보다 여러 대의 로봇이 동시에 용접하는 것이 작업 시간의 감소에 도움을 줄 것이다. 이와 같은 이유로, 최근에 <그림 1>과 같은 서로 평행하게 놓인 4대의 gantry형 용접 로봇으로 이루어진 용접 로봇 시스템이 개발되었다. 그런데 여러 대의 로봇이 동시에 작업하다 보면 인접한 로봇들 사이에 충돌이 발생하는 새로운 문제점이 생기게 된다. 그러나 이러한 문제점은 기존의 로봇을 이용한 전자 부품 조립 공정이나 자동차 조립 공정에서는 발생하지 않는다. 또한, 이 시스템에서 발생하는 '충돌 문제를 고려한 스케줄링'에 대한 기존의 연구결과는 없는 것으로 믿어진다. 따라서, 본 논문에서는 여러 대의 용접 로봇이 서로 간섭을 일으키며 작업을 수행할 때 로봇들 사이의 충돌이 없이 전체 용접 작업의 완료 시간을 최소화하는 스케줄을 작성하는 방법을 제시하고자 한다.

시스템에서 사용하는 로봇의 대수는 변동이 있을 수 있지만 개발된 시스템에서는 4대를 사용하고 있으므로 이를 기준을 하여 알고리듬을 설명하도록 하겠다.

본 논문은 다음과 같은 순서로 구성되어 있다. 2장에서는 대상이 되는 조선 소조립 공정을 설명하고 연구 대상이 되는

문제를 정의하겠다. 3장에서는 해결 방법을 제시하겠으며, 4장에서는 작업 데이터에 대해서 수행한 결과를 분석하겠다. 5장에서는 본 연구의 결과를 정리해서 설명하겠다.

2. 문제 정의

연구의 대상이 되는 조선 소조립 공정은 Array, Mounting, Welding 1, Welding 2, Finishing의 5단계로 이루어져 있다. 첫째, Array 단계에서는 base panel 위에 stiffener를 정해진 위치에 세운다. 다음으로 Mounting 단계에서는 가접, 즉 세워진 stiffener가 넘어지거나 위치가 변하지 않도록 stiffener와 panel 사이를 임시로 용접한다. Welding 1 단계에서는 가접된 상태의 stiffener의 위치를 vision sensor로 인식하여 용접 로봇으로 자동화된 용접 작업을 수행하고, Welding 2 단계에서는 로봇으로 용접할 수 없는 부분이나 주어진 시간 안에 용접하지 못한 부분을 수작업으로 용접한다. 끝으로 Finishing 단계에서는 용접 이후의 후처리를 한다. 각 단계들 사이에서 작업물들은 pallet 위에 놓인 상태로 순서대로 이동한다. 우리가 이 논문에서 살피고자 하는 대상은 Welding 1 단계에서의 용접 자동화이다. Welding 1 단계 공정은 <그림 1>과 같은 로봇 시스템에 의해 수행되며 Z축은 수직축, X축은 가로축, Y축은 세로축을 나타낸다. 그리고 각 용접 로봇의 X, Y, Z축으로의 이동은 각각 독립이다. 다시 말해서 이동시 X, Y, Z축 방향으로 독립적으로 움직이며, X축으로 이동할 때는 v_x의 속도로 이동하고, Y축으로 이동할 때는 v_y의 속도로 이동하며, Z축으로 이동할 때는 v_z의 속도로 이동한다. 따라서, 좌표 (x_1, y_1, z_1)에서 좌표 (x_2, y_2, z_2)까지 최단 거리로 이동할 때 걸리는 시간은

$$\max\left\{\frac{|x_2-x_1|}{v_x}, \frac{|y_2-y_1|}{v_y}, \frac{|z_2-z_1|}{v_z}\right\}$$

이 된다.

각 용접 로봇이 전체 작업 영역을 몇 개의 구간으로 나누어 그 중 한 개의 구간만을 전적으로 용접하는 것이 아니라 작업장 전 영역을 움직이면서 작업한다. 따라서 용접 작업 도중 인접한 로봇들의 X좌표의 차이, 다시 말해 로봇들 사이의 간격이 일정한 값 이상을 유지하지 못할 경우 두 용접 로봇이 충돌하게 된다. 지금부터 충돌이 발생하지 않으면서 인접한 두 용접 로봇이 가장 가까이 접근했을 때, 두 용접 로봇의 중심들 사이의 거리를 충돌 가능 거리라고 부르겠다.

용접 작업은 용접 대상과 방향에 따라 수평 용접과 수직 용접으로 나뉜다. 첫째, 수평 용접은 base panel과 stiffener 사이의 용접으로 base panel 표면을 따라 직선으로 움직이면서 수행하는 용접이며, 용접 도중 높이가 변하지 않기 때문에 Z좌표는 변하지 않고 일정하다. 둘째, 수직 용접은 stiffener와 stiffener 사이의 용접으로 X, Y좌표는 변함없이 Z좌표만 변하면서 수행하는 용접이다. 이렇게 용접 작업을 두 가지로 나눌 수 있지만 보통의 경우 수직, 수평 용접 이외에 수평 용접의 한쪽 또는 양쪽 끝에 수직 용접이 포함된 형태를 하나의 단위로 용접할 수 있다. 본 논문에서는 이러한 하나의 용접 단위 또는 그 것의 일부분을 용접선이라고 정의하겠다.

서론에서 언급하였듯이 stiffener의 양변을 따라 두 개의 용접선이 존재하며 gantry에 달려 있는 두 팔로 동시에 용접한다. 따라서 이 두 용접선의 중심에 가상의 용접선을 생각하고 로봇이 이 용접선을 용접하도록 하는 스케줄은 원래 문제에 대해서도 실현 가능한 스케줄을 제공하기 때문에 본 논문에서는 이러한 가상의 용접선들을 입력받아 이들에 대한 스케줄을 구하는 방법을 제시한다.

용접선 중에서 양 끝점의 X좌표의 차이가 큰 용접선이 존재할 경우 하나의 로봇이 이를 용접한다면 인접한 로봇은 충돌을 피하기 위해서 작업장의 바깥쪽으로 피해 있어야 한다. 다시 말해서 X축으로 긴 용접선이 있어서 이를 하나의 로봇이 용접한다면 그만큼 전체 용접 시간이 늘어날 수밖에 없을 것이다. 이 경우 하나의 용접선을 몇 개의 부분으로 나누고 각각을 여러 대의 로봇이 나눠서 용접한다면 전체 용접 시간을 줄일 수 있을 것이다. 이렇게 하나의 용접선을 여러 부분으로 나눠서 각 부분을 서로 다른 로봇이 용접하는 것을 용접선 분할이라 부른다. 그러나 용접선을 분할할 때 용접의 질을 유지하기 위해서 다음과 같은 두 가지 제약을 지켜야 한다. 첫째, 한 로봇이 용접선을 두 번 이상에 걸쳐 나누어 용접할 수 없도록 하나의 용접선을 여러 개의 부분으로 분할한 경우 두 개 이상의 부분들이 하나의 로봇에 할당되지 않도록 한다. 둘째, 나누어진 각 부분의 길이가 최소한 일정 길이 이상이 되도록 해야 한다. 본 논문에선 분할할 수 있는 최소 길이를 분할 가능 길이라 정의하겠다. 이 논문에선 여러 개로 나뉜 각 부분 또한 용접선이라 부르겠다.

본 연구에서는 실제 구현된 작업장과 같이 4대의 용접 로봇이 사용되는 경우만 다루겠다. 그리고 문제를 정의하면 다음과 같이 나타낼 수 있다.

문제의 환경 변수로는 각 방향의 이동 속도 (v_x, v_y, v_z), 용접 속도 (v_w), 용접 로봇들의 초기 위치, 충돌 가능 거리,

분할 가능 길이가 있다.

그리고 초기 용접선의 집합 $I_0=\{l_1, l_2, \ldots, l_n\}$가 입력으로 들어오며, 용접선의 양 끝점의 좌표 (P_l^L, P_l^R)만이 용접선 l ($\in I_0$)의 정보로 입력된다. 이때, P_l^L은 두 끝점 중 X 좌표가 작은 쪽, 또는 X좌표가 같을 때 Y좌표가 작은 쪽, 또는 Y좌표마저 같을 때 Z좌표가 작은 쪽 끝점의 좌표를 나타내고, P_l^R은 반대편 끝점의 좌표를 나타낸다(앞으로는 X좌표가 작은 쪽을 L(좌측), 큰 쪽을 R(우측)라 하겠다). 용접선은 수직 용접과 수평 용접으로만 이루어져 있기 때문에 끝점의 Z좌표에 따라 수직 용접선의 삽입 여부를 판단할 수 있다. 따라서 이렇게 각 용접선의 두 끝점의 좌표를 알면, 용접선을 정확히 알 수 있고, 용접 속도 v_w를 이용해서 용접 시간(w_l)을 구할 수도 있다.

문제의 목적은 로봇들 사이의 충돌이 없이 전체 작업 시간 T를 최소화하는 용접선의 할당(I_j, $j=1,\cdots,4$)과 용접 스케줄(R_j, $j=1,\cdots,4$)의 생성이다.

설명의 편이를 위해서 각 로봇의 번호를 <그림 1>과 같이 부여하자. 그러면, j번째 용접 로봇의 작업 시간, 즉 시작 시점부터 전체 용접을 끝내는 시각까지의 총 시간을 T_j라고 할 때 전체 작업 시간 T는 $\max_{j=1,\ldots,4} T_j$가 된다.

용접선 할당(I_j)은 각 용접 로봇 j가 용접할 용접선들의 집합 I_j를 결정하는 것으로 I_j ($j=1,\ldots,4$)를 모두 모았을 때 I_0가 되어야 하며, 한 용접선을 하나의 로봇이 두 번 이상 나누어 용접하지 않도록 각 I_j에는 임의의 초기 용접선 l ($\in I_0$)에서 분할된 새로운 용접선이 많아야 한 개씩 포함되어야 한다. 용접 스케줄(R_j)은 각 용접 로봇 j가 I_j에 포함된 용접선들을 용접하는 순서와 용접 방향 그리고 각 위치에 있을 때의 시각을 나타내는 것으로 로봇 j가 용접선 l의 양끝 (P_l^L, P_l^R)에 있는 시각(t_l^L, t_l^R)과 대기 위치(P_d)에 도착하는 시각, 떠나는 시각을 알면 구할 수 있다. 이때 대기(d)란 충돌을 피하기 위해 일정 위치에서 움직이지 않고 멈춰 있는 것을 말하며 용접선과 같은 구조를 이용하여 나타낼 수 있다. 즉, 대기 위치는 P_d^L ($= P_d^R$)로 시작과 끝 시각은 각각 t_d^L, t_d^R로, 대기하는 시간은 w_d로 하는 용접선으로 대기 (d)를 나타낼 수 있다. 그런데 언급하는 용접선 (l)이나 대기 (d)가 많을 때는 이들을 구분하기 위해서 l_1, l_2나 d_1, d_2처럼 아래 첨자를 사용하겠다.

3. 알고리듬

용접 로봇들 사이의 충돌이 없이 전체 작업 시간을 최소화하는 것은 상당히 어려운 문제, 즉 NP-Hard이다. 왜냐하면 단지 한 대로 용접하는 경우에 대해서도 용접 시간의 최소화는 NP-Hard이기 때문이다. 한 대로 용접하는 경우에 대해서 살펴보면 다음과 같다. 용접 로봇은 용접선의 한쪽 끝점에서 출발하여 다른 끝점까지 일정 시간 동안 용접을 한다. 용접 로봇이 한 대이므로 충돌을 피하기 위한 대기 시간이 필요 없고, 용접

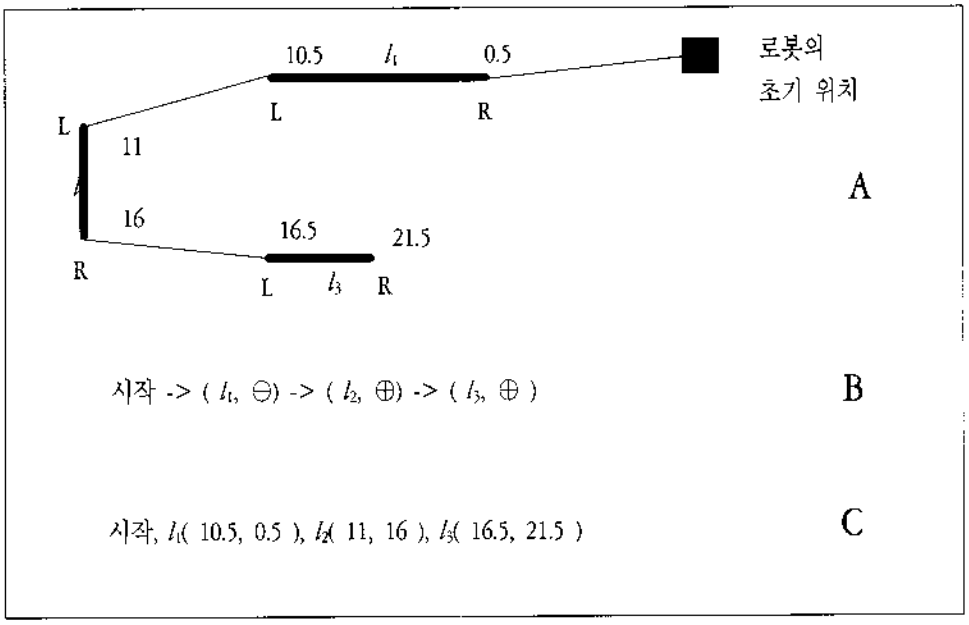

그림 2. 용접 스케줄의 세 가지 표현 방법.

시간이 고정되기 때문에 전체 작업 시간을 최소화하는 것은 이동 시간을 최소화하는 문제와 같아진다.

이러한 문제를 시골 우체부 문제(Rural Postman Problem)라는 이름으로 부르며[10], NP-Hard라는 것이 증명되어 있다[5]. 또한 시골 우체부 문제는 외판원 문제 (Traveling Salesman Problem)로 쉽게 변형될 수 있다. 따라서, 이동시간의 최소화에 외판원 문제에 대한 해법을 이용할 수 있다[8].

로봇이 한 대인 경우 용접 스케줄을 살펴보기 위해서 <그림 2>와 같은 경우를 생각해 보자. A에는 로봇이 지나가는 궤적과 각 용접선의 끝점에 로봇이 위치할 시각이 나타나 있다. B에는 로봇이 지나가는 궤적이 단지 용접선과 용접 방향으로 나타나 있다. 여기서 ⊕는 용접선의 L에서부터 시작해서 R까지 용접하는 것을, ⊖는 용접선의 R에서부터 시작해서 L까지 용접하는 것을 나타낸다. C에는 용접선들이 용접 순서대로 나열되어 있으며, $l(t_L, t_R)$에서 l은 용접선을 t_L은 L에 위치할 시각을 t_R는 R에 위치할 시각을 나타낸다. 다시 말해서 A에 나타난 용접 스케줄은 B나 C의 방법으로 표현할 수 있다. 따라서 이후의 용접 스케줄은 B나 C와 같은 방법으로 나타내겠다.

앞에서 살펴본 바와 같이 다루는 문제가 NP-Hard이다. 또, 용접선 분할 및 할당 등 결정 사항이 많고, 인접 로봇들 사이의 충돌 등 문제의 복잡도를 증가시키는 요소가 많기 때문에 본 연구에서는 문제를 '용접선 할당', '용접 스케줄 결정'이라는 두 가지 문제로 나눠서 각각을 순서대로 해결하는 발견적 해법을 개발하였다. 이를 개략적으로 나타내면 <그림 3>과 같이 나타낼 수 있다. 용접선 할당 문제는 입력된 용접선들을 분할하고 각 용접선을 용접할 로봇을 결정하여 각 로봇 j가

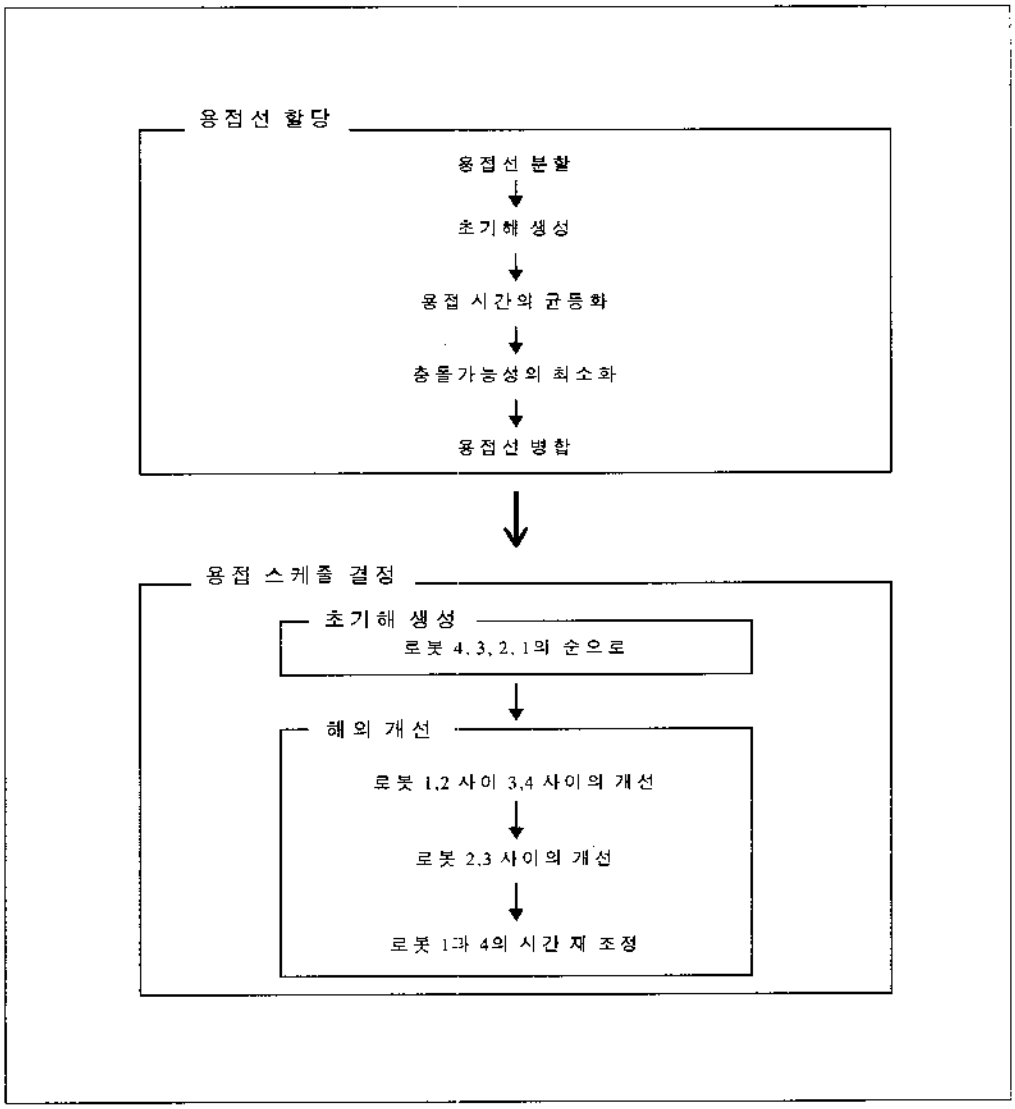

그림 3. 전체 알고리듬의 흐름.

용접할 새로운 용접선들의 집합(I_j)을 생성하는 문제이고, 용접 스케줄 생성 문제는 각 용접 로봇 j가 용접할 용접선들(I_j)이 결정된 상태에서 인접한 용접 로봇들 사이의 충돌이 없이 전체 작업 시간을 최소화하는 용접 스케줄을 결정하는 문제이다.

일반적으로 발견적 해법을 적용하는 경우 위와 같은 두 단계 문제를 더 이상의 개선이 없을 때까지 번갈아 수행한다. 그러나 이 문제에서는 한번 스케줄이 결정된 이후 다시 용접선 할당을 변경한다면 이미 정해진 스케줄이 크게 변하게 된다. 따라서 두 단계를 반복적으로 수행할 경우 해의 개선을 보장할 수가 없고 또한 실현 가능한 해를 구하는 일조차 힘들어지기 때문에 두 단계를 한 번씩만 수행하도록 하였다.

개발된 알고리듬은 <그림 3>과 같다.

3.1 용접선 할당

이동 시간이나 그 밖의 불필요한 시간이 적을 때 각 로봇의 작업 시간(즉, 시작 시점부터 전체 용접을 끝내는 시각까지의 총 시간)의 최대값의 최소화는 전체 로봇들의 총 용접 시간들이 같을 때 이루어진다. 즉, 로봇에 용접선을 할당할 때 각 로봇의 용접 시간을 균등하게 할당한다면 전체 작업 시간의 최소화에 도움이 될 것이다. 또, X좌표상으로 가까운 용접선들이 서로 다른 로봇들에 할당된 경우 이 용접선들을 같은 시점에 용접한다면 용접 로봇들의 충돌 가능성이 높을 것이다.

이 때 충돌이 발생한다면 이를 피하기 위해서 불필요한 대기 시간이 삽입되고, 그만큼 전체 작업 시간은 길어질 것이다. 따라서 용접선을 할당할 때 X좌표상으로 가까운 용접선들을 같은 로봇에 할당하면 충돌 가능성이 그만큼 감소할 것이다. 이러한 면을 고려하여 용접선을 분할하고 할당하는 발견적 해법을 개발하였다.

실제 구현된 해법은 복잡하므로 여기서는 해법의 개요만을 설명하도록 하겠다.

알고리듬 : 할당

입력 초기 용접선의 집합 I_0(각 용접선의 양 끝점들)

출력 용접선 할당 (I_j, $j=1,\cdots,4$)

step 1 초기에 입력된 용접선들을 모두 분할 가능 길이로 나눈다.

step 2 분할 가능 길이로 잘려진 용접선(l)들을 중(P_l^L과 P_l^R의 중심)의 X좌표의 값이 작은 것부터 순서대로 정렬한다 ($\{l_1, l_2, \cdots\}$).

step 3 정렬된 용접선들을 각 용접 로봇에 순서대로 좌에서 우로 할당하여 각 로봇(j)이 용접해야 하는 순수 용접 시간($\sum_{l\in I_j} w_l$)이 비슷한 초기해를 얻는다. 여기서, l_i^j가 로봇 j에 할당된 i번째 용접선이고, 로봇 j에 할당된 부분 용접선의 수가 n_j일 때, 로봇 1부터 3까지 $\sum_{i=1}^{n_j-1} w_{l_i^j} < \frac{1}{4}\sum_{l\in I_j} w_l \le \sum_{i=1}^{n_j} w_{l_i^j}$를 만족하도록 한다.

step 4 $l \in I_{j^*}$를 j^*-1 또는 j^*+1로 재할당하여 최대 용접 시간을 줄인다. 재할당으로 최대 용접 시간을 감소시키는 $l\in I_{j^*}$가 존재하지 않을 때까지 이러한 작업을 반복 수행한다. 이때, $\max_{j=1,\cdots,4}\sum_{l\in I_j} w_l$ 이 최대 용접 시간을, 또 j^*가 최대 용접 시간을 갖는 용접 로봇의 index를 나타낸다.

step 5 충돌 가능성을 최소화하기 위해서 step4에서 얻은 최대 용접 시간을 넘지 않는 한도 안에서 용접선의 할당을 변경하여 X좌표상으로 서로 가까운 용접선들이 같은 용접 로봇에 할당되도록 한다. 자세한 방법은 다양하게 구현될 수 있기 때문에 설명을 생략한다.

step 6 용접선의 할당이 종료되면 원래 같은 용접선에서 분할된 부분들 중에서 같은 로봇에 할당된 것들을 다시 하나의 용접선으로 병합한다.

한 개 용접선에서 분리된 부분 용접선들이 좌에서 우의 순으로 번호가 주어져 있을 때, 각 부분 용접선 i가 할당되는 로봇의 번호를 x_i라고 하자. 그러면 step6을 수행하기 위해서는 step5의 결과가 $i<j$ 인 부분 용접선들에 대해서 $x_i \le x_j$를 만족해야 한다.

이는 충돌 가능성과 하나의 용접선이 한 로봇에 의해 두 번 이상에 걸쳐 나뉘어 용접될 수 없다는 제약에 의한 것이다. 따라서 이러한 제약에 맞도록 step4와 step5가 설계되어야 한다.

3.2 용접 스케줄 결정

여기서는 3.1에서 구한 할당(I_j, $j=1,\cdots,4$)에 대해 용접 로봇들 사이의 충돌 없이 전체의 작업 시간이 최소가 되는 용접 스케줄 (R_j, $j=1,\cdots,4$)을 구하는 것을 목적으로 하며, 초기해의 생성 단계와 이의 개선 단계로 나눠서 최종 용접 스케줄을 생성한다.

3.2.1 초기해의 생성

초기해 생성 과정에서는 3.1에서 구한 할당( ) 에 대해 용접 로봇들 사이의 충돌 없이 전체의 작업 시간이 최소가 되도록 각 용접 로봇별로 순차적으로 용접 스케줄을 생성한다. 즉, 로봇 4의 용접 스케줄을 결정한 이후에 로봇 3과 로봇 4의 충돌이 없도록 로봇 3의 용접 스케줄을 결정하고, 로봇 2와 로봇 3의 충돌이 없도록 로봇 2의 용접 스케줄을 결정하고, 마찬가지 방법으로 로봇 1의 용접 스케줄을 결정한다.

따라서, 로봇 4를 제외한 로봇들의 용접 스케줄을 결정할 때는 우측의 용접 로봇과의 충돌만을 고려하면 된다. 이러한 스케줄링에서는 한 시점의 결정 변수가 줄고 충돌 회피가 간단하게 되는 장점이 있다.

용접 로봇들 사이의 충돌을 줄이기 위해서 로봇들이 기본적으로 따라야 하는 기본 이동 방향을 정의할 수 있다. 즉, <그림 4>처럼 전체 용접 로봇들이 점차적으로 동일한 방향으로 이동하면서 작업을 한다면 인접한 로봇들 사이에서의 충돌 가능성은 그 만큼 줄어들 것이다. 이러한 기본 이동 방향을 구현하기 위해서, 한 용접 로봇 j에 할당된 용접선의 집합 I_j을 <그림 5>에서처럼 4가지 집합(A_j, B_j, C_j, D_j)으로 나눈다(참고로 로봇 1의 경우 C_1과 D_1만 존재하고 로봇 4의 경우 A_4와 C_4만 존재한다). 이때, α는 로봇 $j-1$의 이동 영역 중 로봇 중심의 오른쪽 한계 위치에서 충돌 가능 거리만큼 떨어진 지점이다. 따라서, 로봇 j의 중심이 α의 우측에 위치할 경우에는 로봇 $j-1$과 충돌하지 않을 것이다. 마찬가지로 β 또한 로봇 $j+1$의 이동 영역 중 로봇 중심의 왼쪽 한계 위치에서 충돌 가능 거리만큼 떨어진 지점으로 로봇 j의 중심이 β의 왼쪽에 위치할 경우 로봇 $j+1$과 충돌하지 않는다. 따라서, 두 끝점의 X좌표가 모두 α보다 큰 용접선들을 용접

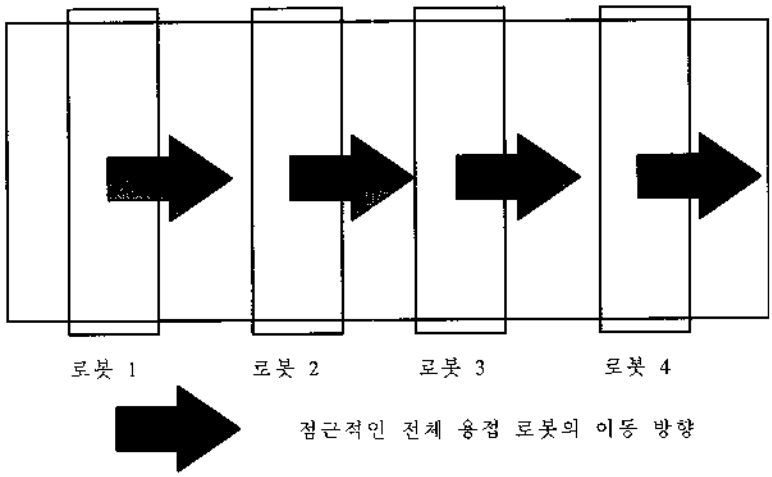

그림 4. 기본 이동 방향.

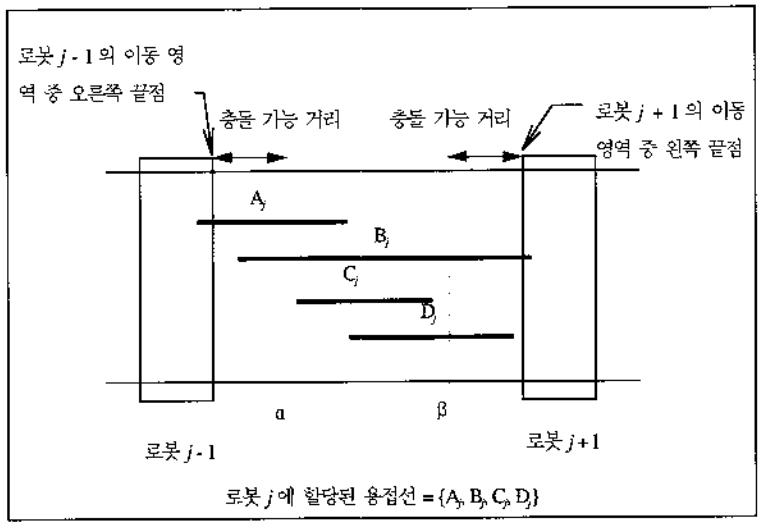

그림 5. 용접선의 분류.

할 때는 어떤 경우든 용접 중인 $j-1$번째 용접 로봇과 충돌을 일으키지 않는다. 마찬가지로 두 끝점의 X좌표가 모두 β보다 작은 용접선들은 용접중인 $j+1$번째 로봇과 충돌을 일으키지 않는다. 그러므로 집합 A_j는 용접중 로봇 $j-1$과 충돌을 일으킬 수 있는 용접선들의 집합이고, 집합 B_j는 용접중 로봇 $j-1$과 로봇 $j+1$ 모두와 충돌을 일으킬 수 있는 용접선들의 집합이고, 집합 C_j는 용접중 충돌을 일으킬 가능성이 없는 용접선들의 집합, 그리고 집합 D_j는 용접중 로봇 $j+1$과 충돌을 일으킬 수 있는 용접선들의 집합을 나타낸다. 이렇게 구분된 집합들을 순차적으로 용접하면 로봇들은 결과적으로 앞에서 보인 <그림 4>의 기본 이동 방향으로 움직이게 될 것이다. 그런데 충돌 가능 거리가 작업장의 크기에 비해 상대적으로 긴 경우엔 α와 β사이의 폭이 좁아져 용접 중 인접한 로봇들과 충돌을 일으킬 수 있는 용접선이 많아지게 된다. 그 결과 용접선들이 같은 집합 안에 존재하더라도 용접 중 충돌을 일으키는 정도가 많은 차이를 보일 수 있다. <그림 6>에서처럼 D_j가 9개의 용접선으로 이루어져 있을 때, l_1과 l_9를 비교해 보면 모두 D_j에 포함되어 있지만 l_1이 l_9에 비해서 확실히 로봇 $j+1$과 충돌할 가능성이 높다. 따라서 D_j를 <그림 6>에서처럼 (D_j^1, D_j^2, D_j^3, D_j^4)으로 세분하고 이를 순차적으로 용접한다면 <그림 4>에서 나타난 기본 이동 방향으로 더욱 세밀히 움직이게 되며 따라서 인접 로봇과의 충돌 가능성을 그 만큼 감소시킬 것이다. 이렇게 용접선의 집합을 세밀하게 분류하는 방법은 여러 가지로 구현될 수 있으나, 여기서는 오른쪽 로봇에 할당된 용접선 중 충돌 가능 거리 이내에 분포하는 것의 개수에 따라 분류하였다.

하나의 세분화된 용접선의 집합에 포함된 용접선들의 충돌 가능성이나 형태는 서로 유사하다. 즉, <그림 6>에서 D_j^3에 포함된 l_4, l_5, l_6를 살펴보면, 특정 시점에 l_4를 용접하는 것이 충돌을 유발한다면 그 시점에 l_5나 l_6를 용접하는 것도 마찬가지로 충돌을 유발한다. 따라서, D_j^3에 포함된 용접선들 사이에서 용접 순서를 결정할 때에는 로봇 $j+1$과의 충돌을 고려하지 않고 단지 이동 시간을 최소화하는 용접 순서를 결정한 후 D_j^2에 속한 용접선들의 작업을 끝낸 시각과 로봇의 위치 그리고 로봇 $j+1$의 용접 스케줄에 맞춰서 로봇 $j+1$과 충돌하지 않도록 D_j^3에 포함된 용접선들의 용접 시간대와 용접 방향을 결정해 준다. 그런데 연속해서 용접하는 두 용접선들에 대해서, 먼저 용접하는 용접선의 용접이 종료되는 시각과 다음 용접이 시작되는 시각 사이의 차이가 두 지점(용접이 끝난 지점과 시작되는 지점) 사이의 이동 시간과 달라지는(보다 커지는) 경우가 발생한다. 이러한 경우 로봇은 이

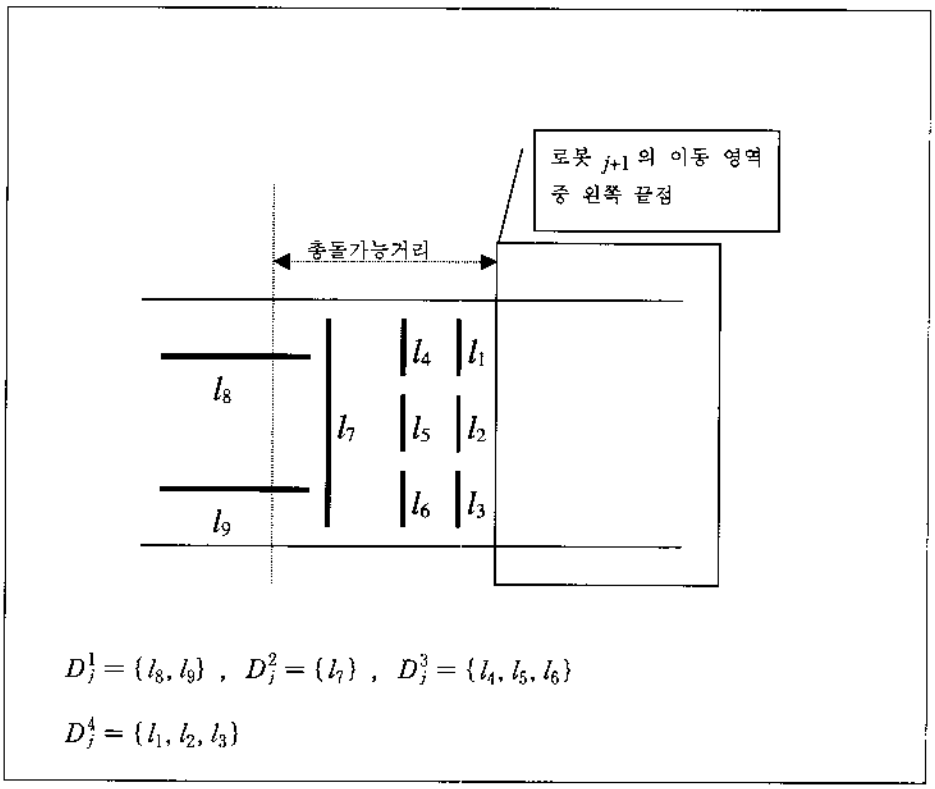

그림 6. 용접선의 세분화.

동에 소요되는 시간을 제외한 시간 동안 인접 로봇과 충돌하지 않는 위치에서 대기하여야 한다. 대기 위치는 충돌을 고려하여 두 용접선의 용접 종료와 시작 사이의 시간 안에 이동할 수 있는 위치로 결정할 수 있다.

용접 순서 결정시에는 시작점(용접 시작 전 로봇의 초기 위치)에서 출발하여 대상이 되는 용접선을 용접하는 시간을 최소화하는 용접 경로를 구한다. 물론 여기서 구한 용접 경로는 충돌을 고려하지 않고 만들어지기 때문에 용접 스케줄에 그대로 반영되지 않고 단지 용접 순서만이 반영된다.

용접 순서 결정 문제는 시골 우체부 문제라고 불리는 변형된 외판원 문제로 나타낼 수 있다. 시골 우체부 문제를 나타내는 값들로는 그래프 $G=(N,E)$ (N은 노드 집합, E는 edge집합)와 각 edge $e\in E$의 비용 c_e, 그리고 꼭 지나야 하는 edge들의 집합 $R\subseteq E$ 이 있다. 시골 우체부 문제는 주어진 graph G에서 정해진 edge의 집합 R에 포함된 모든 edge를 지나는 최소 비용의 cycle을 구하는 문제이다.

용접 순서 결정 문제는 시작점에서 출발하여 대상 용접선을 모두 지나는 최단 경로를 구하는 문제이다. 그러나 시골 우체부 문제는 cycle을 이뤄야 하고 시작점은 반드시 지나야 하기 때문에 v^*라는 가상의 노드와 이 노드와 시작점을 잇는 새로운 edge e^*를 도입하여 e^*를 꼭 지나야 하는 edge의 집합 R에 포함시킴으로써 용접 순서 결정 문제를 시골 우체부 문제로 변환할 수 있다. 다음은 시골 우체부 문제를 나타내는 환경값들을 정하는 방법에 대한 설명이다.

노드의 집합 N은 n개의 용접선의 양끝에서 정의되는 $2n$개의 끝점과 로봇의 초기 위치, 그리고 v^*로 정의된다. 로봇의 초기 위치는 작업이 시작할 경우 로봇의 작업 시작 위치이고 그렇지 않을 경우 바로 직전에 작업이 끝나는 위치(용접선의 끝점)가 된다. 로봇의 작업 시작 위치는 현장의 사정에 따라 달라지겠지만 로봇 1, 2의 시작 위치는 (0, 0, 0)으로 로봇 3, 4의 초기 위치는 (작업장의 폭, 0, 0)으로 정하였다. 용접선의 수가 n개일 때 N의 원소의 수는 $2n+2$가 된다. E는 N에 포함된 모든 노드 사이에 정의된다. 그리고 단순 이동을 나타내는 edge $e\in E\backslash R$ 중 v^*와 연결되지 않는 edge의 비용 c_e는 두 끝점 사이의 이동 시간으로 정하며 v^*와 연결된 edge의 c_e는 0으로 한다. 또, 꼭 지나야 하는 edge $e\in R-\{e^*\}$의 비용 c_e는 그 edge에 대응되는 용접선의 용접 시간으로 하며, c_{e^*}는 0으로 한다.

시골 우체부 문제는 변형된 외판원 문제로 NP-hard임이 증명되어 있다. 또, 이 문제는 외판원 문제와 유사한 방법으로 접근이 가능하기 때문에 외판원 문제를 위한 발견적 해법을 일부 변형하여 이 문제에 적용할 수 있다. 외판원 문제에 대한 발견적 해법으로 한 번에 해를 생성하는 방법과 기존 해의 개선 방법들이 연구되어 있다[4][7]. 이 중 본 논문에서는 farthest insertion을 통한 초기해의 생성과 two-opt exchange 방법을 통한 해의 개선 방법을 사용하였다[4][6].

Farthest insertion 방법은 단지 작은 집합의 노드만을 지나는 cycle에서 시작하여 남은 노드들을 삽입함으로써 이 cycle을 확장하는 방법이다. 이때 노드의 삽입은 이 cycle에서 가장 멀리 떨어진 것부터 한다. 또 two-opt exchange 방법은 cycle에 포함된

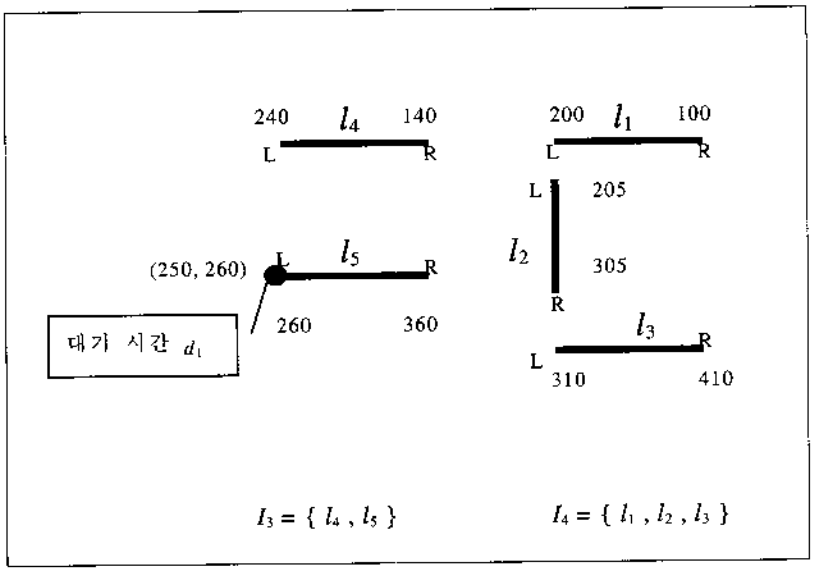

그림 7. 용접 시간대의 예.

임의의 두 edge를 삭제하여 분리된 두 개의 경로를 cycle에 나타나지 않는 두 edge를 삽입함으로써 하나의 cycle로 만드는 방법으로 개선을 이룰 수 있는 edge들에 대해서 반복해서 수행한다. 이 두 방법을 용접 순서 결정 문제에 사용하기 위해서 farthest insertion의 경우 작은 수의 R에 속하는 edge들을 포함하는 cycle에서 가장 멀리 위치한 노드에 관계된 edge $e \in R$를 cycle에 삽입한다. Two-opt exchange의 경우 삭제할 edge의 대상을 $E \backslash R$의 원소로 제한한다.

용접 작업 중 인접한 두 로봇들 사이의 충돌을 피하기 위해서 다음과 같은 상황을 생각해 보자. 두 용접선 $l \in I_j$, $l' \in I_{j+1}$의 X축상의 거리가 충돌 가능 길이보다 가깝다고 생각하자. 그러면, 두 로봇 j와 $j+1$이 l과 l'을 용접할 때 충돌을 피하기 위해서 두 용접 작업 사이에 일정한 시간차를 두어야 할 것이다. 그런데 이러한 '충돌을 피하기 위해 고려해야 하는 시간차'는 용접 방향과 어떤 것을 먼저 용접하느냐에 따라 8가지($l \oplus \oplus$, $l \oplus \ominus$, $l \ominus \oplus$, $l \ominus \ominus$, $l' \oplus \oplus$, $l' \oplus \ominus$, $l' \ominus \oplus$, $l' \ominus \ominus$)가 존재한다. 여기서 첫째 값은 먼저 용접을 시작하는 용접선을, 두 번째 값은 l의 용접 방향을, 마지막 값은 l'의 용접 방향을 나타낸다. 따라서 $l \in I_j$의 용접 시간대와 용접 방향의 결정은 로봇 $j+1$의 용접 스케줄에 대해서 정해지는 l의 용접을 시작할 수 있는 시간대들 중 적당한 시점에 l을 할당하는 것과 같다. 이때, 작업을 빨리 하기 위해서 용접을 시작할 수 있는 시간대들 중에서 가장 빠른 시점을 선택한다. 여기서 말하는 용접을 시작할 수 있는 시간대들은 로봇 $j+1$의 스케줄과 l의 용접 방향에 따라 l의 용접을 시작할 수 있는 시간대들을 말한다. 따라서, 로봇 j가 용접해야 하는 순서가 결정되면 이 순서대로 각 용접선을 용접할 방향과 시간대를 구할 수 있다. 위에서 언급한 내용으로 알고리듬을 다음과 같이 구현하였다.

알고리듬 : 초기 용접 스케줄 결정

입력 용접선 할당 (I_j, $j=1,\cdots,4$)

출력 각 로봇의 스케줄 (R_j, $j=1,\cdots,4$)

step 0 $R_1 = R_2 = R_3 = R_4 = \emptyset$, $j=4$

step 1 I_j의 용접선들을 분류, 정렬해서

$J^* = \{A_1, \cdots, A_{a_j}, B_1, \cdots, B_{b_j}, C_1, \cdots, C_{c_j}, D_1, \cdots, D_{d_j}\}$를 만든다.

(여기서 a_j, b_j, c_j, d_j는 I_j 중 A, B, C, D에 속하는 부분 집합의 수를 의미한다.)

step 2 $J \Leftarrow J^*$의 첫 번째 부분집합, $J^* = J^* - J$

step 3 J의 용접선들 사이의 용접 순서 결정(충돌을 고려하지 않음)

step 4 step 3에서 구한 용접 순서대로 용접 스케줄 결정(용접 시간대, 용접 방향, 대기 시간)하여 R_j의 후미에 첨가한다.

step 5 $J^* \neq \emptyset$이면 step 2로

step 6 $j \geq 2$이면 $j = j-1$, step 1로 아니면 완료.

생성된 초기해의 예를 그림을 통해서 설명하면 <그림 7>과 같다. <그림 7>에는 로봇 3과 로봇 4의 용접 스케줄이 나타나 있다. 로봇 4의 용접 스케줄(R_4)을 미리 결정하고, 로봇 3의 용접 스케줄(R_3)은 로봇 4와 충돌이 일어나지 않도록 결정하였다. <그림 7>에 나타나는 d_1은 로봇 3이 로봇 4와 충돌하지 않도록 동작을 정지하는 것을 나타낸다. 이를 이해하기 위해서 로봇 3의 스케줄을 살펴보면, 로봇 3이 $P^L_{l_2}$(<그림 7>에서 각 용접선 l에 나타나는 L과 R는 P^L_l과 P^R_l을 의미함)에 시점 250에 도착하였으나 시점 250에 용접을 시작하면 l_2를 용접하는 로봇 4와 충돌할 수 있기 때문에, 시점 250($t^L_{d_1}$)에서부터 시점 260($t^R_{d_1}$)까지 대기하고 260($t^L_{l_2}$)부터 용접을 시작한다. 이와 같이 로봇 $j+1$과 로봇 j 사이의 충돌은 로봇 j의 스케줄에 대기 시간을 포함시킴으로써 회피할 수 있다.

3.2.2 해의 개선

3.2.1에서 구한 초기해가 만족스럽지 못한 경우 작업 순서를 고정하고 용접 방향과 용접 시간대를 변경해서 해의 개선을 이룰 수가 있다. 이 때도 전체를 한 번에 개선하는 것은 어렵기 때문에 인접한 두 로봇들 사이에서 개선을 이루고 이를 이용하여 전체의 개선을 이룬다. 그럼 먼저 인접한 두 로봇들 사이에서의 해의 개선을 설명하고 이를 전체적으로 적용하는 방법을 설명하겠다.

$I_j(=\{l_1, l_2, \ldots, l_{n_j}\})$ 와 $I_{j+1}(=\{l'_1, l'_2, \ldots, l'_{n_j}\})$의 용접선들이 용접되는 순서대로 정렬되어 있을 때 로봇 $j-1$이나 $j+2$와는 상관없이 단지 로봇 j와 $j+1$의 용접 시간을 최소화하는 용접 스케줄을 생성하는 방법은 다음과 같다[1].

1) 이때, I_j와 I_{j+1}에는 앞에서 언급된 용접선들 이외에 유지될 필요가 있는 대기 시간도 포함될 수 있으며, 필요에 따라 각 용접선의 용접 방향 또한 고정될 수 있다.

단지 두 로봇의 스케줄만 고려한 경우 하나의 완벽한 스케줄 S를 용접선과 이들의 용접 방향을 용접 순서대로 나열함으로써 표현할 수 있다. 예를 들어 <그림 7>에 나타난 스케줄을 $\{(l_1,\ominus),(l_4,\ominus),(l_2,\oplus),(l_5,\oplus),(l_3,\oplus)\}$로 나타낼 수 있다. 이 때, l_2는 시점 205부터 용접이 시작되고, l_5는 시점 260부터 시작하기 때문에 스케줄에서 l_2가 l_5보다 먼저 나타난다. 또한 l_2를 용접할 때 L에서 R 방향으로 용접하기 때문에 용접 방향은 $\oplus$가 된다.

이와 같은 방법으로 $l_1,\cdots,l_{k_1},l'_1,\cdots,l'_{k_2}$로 이루어진 부분 용접 스케줄[2] $S(k_1,k_2)$를 나타낸다고 생각하자. 각 $S(k_1,k_2)$에 대해서 두 용접선 l_{k_1+1}, l'_{k_2+1} 중 어떤 것을 $S(k_1,k_2)$에 먼저 포함시켜 먼저 용접을 시작하게 하느냐에 따라 두 개의 부분 스케줄 $S(k_1+1,k_2)$과 $S(k_1,k_2+1)$가 생성된다.

그런데 부분 스케줄 $S(k_1,k_2)$가 결정된 상태에서는 그 이후에 결정되는 용접선의 스케줄을 어떻게 정해도 얻을 수 없는 총 용접 시간(두 로봇 모두 작업이 끝나는 시각)의 이론적 하한값(v_S)을 정할 수 있다. v_S는 $\{S(k_1,k_2)$ 에서 로봇 j의 끝 시각 $+\sum_{i\in R_{k_1+1}}^{n_1} w_i\}$와 $\{S(k_1,k_2)$에서 로봇 $j+1$의 끝 시각 $+\sum_{i\in R_{k_2+1}}^{n_2} w_{i'}\}$ 중 큰 값으로 정할 수 있다.

이와 같은 내용을 바탕으로 로봇 j와 $j+1$만을 고려한 스케줄링을 분지 한계법[3][9]를 이용하여 구현하였고 그 알고리듬은 다음과 같다.

알고리듬 : 개선을 위한 분지 한계법

입력 인접한 두 로봇(j, $j+1$)의 초기 스케줄에 따라 정렬된 용접선의 집합(I_j, I_{j+1})

출력 두 로봇(j, $j+1$)의 용접 스케줄 ($S = R_j \bigcup R_{j+1}$)

step 0 v를 이미 알고 있는 값 또는 ∞로 정한다.[3]

step 1 부분 용접 스케줄을 저장할 리스트 L을 만든다. 이론적 하한값을 ∞로 하고, $S(0,0)$을 에 삽입한다.

step 2 L에서 이론적 하한값 v_S가 가장 작은 S를 선택한 후 S를 L에서 제거한다.

step 3 $v_S > v$ 이면 이 알고리듬으로는 작업시간이 v보다 작은 스케줄을 구할 수 없다. 끝. $v_S \le v$ 이면 step 4로 간다.

step 4 k_1와 k_2를 S의 두 인수값으로 지정한다. k_1이 n_1이고 k_2가 n_2이면 S가 최적 스케줄이다. 끝. 그렇지 않다면 step5로 간다.

step 5 l_{k_1+1}를 S에 삽입하고 용접 작업이 빨리 시작하는 방향으로 용접 방향[4]과 용접 시간대를 결정하여 $S^1 = S(k_1+1,k_2)$를 만들고 v_{S^1}을 계산한 후 L에 삽입한다. l'_{k_2+1}을 S에 삽입하고 용접 방향과 용접 시간을 결정하여 $S^2 = S(k_1,k_2+1)$을 만들고 v_{S^2}를 계산한 후 L에 삽입한다. step2로 간다.

사실 최적해를 얻기 위해선 위 알고리듬처럼 l_{k_1+1}, l'_{k_2+1} 중 어느 것의 용접을 먼저 시작할 것인가를 결정할 뿐만 아니라 용접 방향 또한 결정하여 각 단계에서 4가지의 부분 용접 스케줄을 만들어야 하지만 너무 많은 부분 스케줄들이 생성되기 때문에 먼저 용접할 용접선의 선택에 따라 단지 두 개의 부분 스케줄들을 생성하고 각 용접 방향은 초기해의 결정에서 언급한 '충돌을 피하기 위한 시간차'를 고려해서 용접을 빨리 시작할 수 있는 방향으로 정한다. 이때는 이동시간 뿐 아니라 인접 로봇과의 충돌 가능성도 고려한다.

인접한 두 로봇들 사이의 개선을 통해서 전체의 개선을 이루는 알고리듬을 개발할 때 고려해야 하는 사항으로는 전체 작업 시간의 최소화, 두 로봇들 사이의 개선 횟수의 최소화를 들 수 있다. 여기서 두 로봇들 사이의 개선 횟수의 최소화란 앞에서 기술한 개선을 위한 분지 한계법이라는 알고리듬을 수행하는 횟수를 줄인다는 의미이다. 이를 위해서 여기서는 아래와 같은 알고리듬을 제시하였으며 이 알고리듬에서는 두 로봇들 사이의 개선을 3번, 스케줄 변경을 2번 수행한다. 저자의 판단으로는 4대의 로봇이 사용되는 경우 이 알고리듬이 위의 두 기준으로 볼 때 최적에 가까울 것으로 생각한다. 제안하는 알고리듬은 다음과 같다.

알고리듬 : 전체 개선

입력 각 용접 로봇들의 초기 스케줄 (R_j, $j=1,\cdots,4$)

2) 부분 용접 스케줄이란 시작 시점부터 특정 시점까지의 작업을 나타내는 용접 스케줄이다.

3) 로봇 1과 2만을 고려한 개선이나 로봇 3과 4만을 고려한 개선시 v는 초기해에서 두 용접 로봇의 작업 종료 시간의 최대값이 된다. 그러나 해의 개선 과정에서 스케줄이 초기해와 달라지는 경우에는 v는 ∞가 된다.

4) 로봇 2와 3 사이의 개선 과정처럼 용접 방향이 고정된 경우는 이를 유지한다.

출력 개선된 전체 스케줄

step 1 로봇 1과 2만을 고려한 스케줄의 개선 과정을 수행한다. 로봇 3과 4만을 고려한 스케줄의 개선 과정을 수행한다.

step 2 I_2, I_3의 용접 방향과 포함된 대기 시간을 유지하면서 로봇 2와 3만을 고려한 스케줄의 개선 과정을 수행한다.

step 3 로봇 2의 스케줄에 맞춰서 로봇 1의 스케줄을 조정한다. 로봇 3의 스케줄에 맞춰서 로봇 4의 스케줄을 조정한다.

step 2의 용접 방향과 대기 시간을 유지하기 위한 방법은 다음과 같다. '개선을 위한 분지 한계법'의 step5에서 고정된 용접 방향을 유지하도록 함으로써 용접 방향의 고정을 해결할 수 있으며, 2장에서 언급한 것처럼 양 끝점의 좌표를 대기하는 좌표로, 용접 시간을 대기하는 시간으로 하는 가상 용접선을 스케줄과 I_j에 포함시킴으로써 대기 시간의 유지를 이룰 수 있다.

이 알고리듬을 해의 성능 측면에서 살펴보자.

인접한 두 로봇들 사이에서의 개선이 원활히 이루어질 때, step1과 step2는 원활히 수행될 것이다. 그러나 step3을 원활히 수행하기 위해서는 단순히 인접한 두 로봇들 사이에서의 개선 이외의 추가적인 사항, 즉 로봇 1과 로봇 4의 작업 시간의 증가량을 고려해야 한다. 다시 말해서 로봇 2와 로봇 3의 스케줄에 맞춰 로봇 1과 로봇 4의 스케줄을 재결정할 때, 로봇 1과 로봇 4의 작업 시간의 증가량이 각각 로봇 2와 로봇 3의 작업시간의 증가량에 비해 크게 증가하지 않도록 해야 한다. 이렇게 하기 위해서 다음과 같은 두 가지 점을 고려해 보자. 첫째, step1에서 구한, 스케줄 S에 나타나는 로봇 1과 로봇 2의 용접 순서와 용접 방향이 유지될 필요가 있다(아래의 설명은 로봇 3과 로봇 4의 경우에도 마찬가지로 적용된다). 이를 위해서 step2에서 로봇 2와 로봇 3 사이의 개선을 할 때 용접 방향과 대기를 유지하도록 하였다. 둘째, 로봇 2의 스케줄에 로봇 1의 작업을 방해하는 새로운 대기가 발생하지 않아야 한다. 이해를 돕기 위해 다음과 같은 상황을 생각해 보자. 스케줄 S에서 특정 시점에 로봇 2가 $l_1(\in I_1)$을 용접하는 로봇 1을 방해하지 않도록 대기(d_1)한 이후에 $l_2(\in I_2)$를 용접하도록 되어있다. 그런데 로봇 2가 step2에서 로봇 3과의 충돌을 피하기 위해 d_1에 이어 대기 위치 P_{d_1}보다 좌측에서 일정 시간 대기(d_2)하고 나서 l_2를 용접한다. 그 결과 step3에서 로봇 1이 l_1을 용접하는 시점은 로봇 2의 d_2 대기 시점과 l_2 용접 시점 이후가 될 수 있을 것이다. 그 결과 로봇 1의 l_1 이후 용접 작업은 로봇 2의 용접 작업과 어긋나게 되어 S에 나타나는 용접 순서와 크게 달라질 수 있으며 그 결과 로봇 1의 작업 시간을 극도로 증가시킬 수가 있다. 따라서, 이러한 경우가 발생하지 않기 위해서는 로봇 3의 위치가 좌로 치우치는 경우 즉, I_3과 I_4의 분포 영역이 좌로 치우치는 경우가 없어야 한다. 그런데 여러 실험 데이터에 대해 수행한 결과 I_3과 I_4의 분포 위치가 좌로 치우친 경우에도 작업 시간이 극도로 증가하는 경우는 발생하지 않았다. 또한 로봇 3이 좌로 치우침으로써 로봇 1의 작업 시간을 증가시키는 경우가 발생하지 않는다면, 로봇 1의 작업 시간의 증가량이 로봇 2의 작업 시간의 증가량에 비해 크지 않게 된다. 따라서, 이 경우 step2에서 로봇 2와 로봇 3의 작업 시간을 최소화하였다면, 전체 작업 시간의 최소화도 이룰 수 있게 된다.

4. 실험 결과

제시된 알고리듬의 성능을 알아보기 위해서 프로그램으로 구현하여 실험하였다. 실험에서는 실제 구현된 작업장과 비슷한 비율로 폭(X)을 20m, 길이(Y)를 10m, 높이(Z)를 2m, 충돌 가능 거리를 2m, 최소 분할 가능 길이를 400mm, 용접 속도를 10mm/초, X축의 이동 속도를 400mm/초, Y축의 이동 속도를 400mm/초, Z축의 이동 속도를 100mm/초로 정하였다.

알고리듬의 성능을 평가하기 위해서 현장에서 제공된 31개 예제 데이터에 대해서 프로그램을 수행하였다. 이들 예제 데이터에서 총 순수 용접 시간은 평균 4,566초, 최소 2,024초, 최대 11,655초였다. 그리고 해를 평가하기 위해서 다음과 같이 성능척도 A와 B를 정의하겠다.

$$A = \frac{\text{총 순수 용접 시간}}{4 \times \text{전체 작업이 끝나는 시각}}$$

$$B = \frac{\text{각 } \textit{Gantry robot}\text{의 작업이 끝나는 시각의 평균}}{\text{전체 작업이 끝나는 시각}}$$

A는 총 작업 시간과 작업 시간의 이론적 하한값의 비율을 나타내며, 알고리듬의 직접적인 효율을 나타내는 것으로 1에 가까울수록 전체 작업 시간이 작아지고, B는 총 작업 시간과 각 로봇의 작업 시간의 평균의 비율을 나타내며, 값이 1에 가까울수록 전체 로봇이 비슷한 시각에 종료한다. 그런데 구현된 프로그램에서는 용접 순서 결정 단계의 초기해의 A 값이 0.9 미만일 때 해의 개선을 시도하였다.

표 1. 31개 데이터의 최종해의 성능 척도

		A			B	
	최소	평균	최대	최소	평균	최대
초기해	0.580	0.837	0.982	0.709	0.909	0.996
최종해	0.763	0.914	0.982	0.868	0.964	0.996

표 2. 해의 개선과정을 수행한 18개 데이터의 각 단계별 결과 분석

	A			B		
	최소	평균	최대	최소	평균	최대
초기해	0.580	0.755	0.879	0.709	0.863	0.929
최종해	0.763	0.888	0.950	0.927	0.960	0.986
차이	0.039	0.133	0.316	0.040	0.097	0.256

표 3. 해의 개선 과정을 수행한 18개 데이터의 각 단계별 수행 시간(초)

		최소	평균	최대
할당단계		0.27	0.55	1.21
순서 결정 단계	초기해	0.11	0.33	0.71
	해의 개선	0.00	0.39	3.68
전체		0.55	1.27	4.83

C언어로 구현된 프로그램을 PC(Pentium,166MHz)에서 실행하여 31개 예제 데이터에 대한 해를 생성하였다. <표 1>은 생성된 해의 성능 척도 A, B 값을 나타낸다. <표 1>에서 '초기해' 항목이 나타내는 것은 초기해의 성능값이고, '최종해' 항목이 나타내는 것은 해의 개선 과정을 포함한 전체 수행 결과의 성능값이다. 전체 작업의 성능값의 평균이 0.914와 0.964로 만족스러운 결과라고 할 수 있을 것이다. 프로그램의 수행시간은 평균 1.43초, 최소 0.33초, 최대 5.38초로 상당히 빠른 시간임을 알 수 있다. 전체 31개 예제 데이터 중 18개가 해의 개선 과정을 수행하였고, <표 2>는 이들 데이터에 대한 초기해와 최종해의 성능값을 보여준다. 성능척도 A의 평균값이 0.755에서 0.888로 0.133의 개선을 이루었고 성능 척도 B의 평균값이 0.863에서 0.960으로 0.097의 개선을 이루었다. <표 2>의 '차이' 항목은 개선 과정 전, 후 성능 척도의 변동값을 나타낸다. 개선 과정을 수행하는 기준이 초기해의 결과가 좋지 못한 것이라는 것을 염두에 둘 때, 좋지 못한 예제에 대해서도 만족할 만한 결과를 얻었다고 볼 수 있을 것이다. <표 3>은 이들 18개 데이터의 각 단계의 수행 시간을 나타낸다. 평균적으로 각 단계의 수행 시간이 비슷하지만 최대값을 보면 해의 개선 단계의 수행 시간이 상당히 크다는 것을 알 수 있다. 이는 사용된 분지 한계법의 수행 시간이 입력된 용접선의 수에 지수적으로 비례하기 때문이다.

5. 결 론

우리는 본 논문에서 조선 소조립 공정에 사용되는 자동화된 용접 로봇 시스템을 위한 용접 스케줄링 문제를 정의하였고, 이 문제에 대한 알고리듬을 개발하였다. 그리고, 알고리듬의 성능을 평가하기 위해서 현장에서 제공된 데이터에 대해서 스케줄링하였고, 만족할 만한 결과를 얻었다.

사용된 알고리듬은 용접선 할당과 스케줄링이라는 두 단계로 이루어지며, 용접선 할당 단계에서는 용접선 분할, 스케줄링에서는 로봇의 기본 이동 방향이 도입되어 로봇들 사이의 충돌 회피를 용이하게 하였다. 그런데 현장에서의 로봇 작동 시 예기치 못한 오작동이 발생할 수 있으며, 로봇 작동 시간이 예측한 것과 달라질 수 있다. 그 결과로 로봇의 용접과 이동 시간이 스케줄과 달라져 인접 로봇들 사이의 충돌을 야기할 수 있다. 그러나 이러한 경우 로봇에 달린 센서를 통해 컨트롤러가 충돌을 인식하여 로봇 작업을 일시 정지시킬 수 있다. 그렇기 때문에 현장에서는 이러한 정지 상태가 발생할 경우 남은 작업에 대한 새로운 스케줄을 생성하고 이에 따라 작업을 재개하는 방법을 사용하고 있다. 따라서, 본 연구에서 제시한 알고리듬의 성능을 높이기 위해서는 로봇 동작을 정확히 예측하여 로봇의 작동 시간을 알고리듬에 반영할 필요가 있을 것이다.

참고문헌

1. 김진오, 신정식, 김성권, "로봇을 이용한 조선소조립 용접 자동화 시스템," *KAAC 11th 추계 기계학술 대회 Proceeding*, pp. 516-519, 1996.
2. 김진오, "FMS에 있어서 로봇의 역할과 응용," 제어계측, pp. 9-17, 1996. 8.
3. Baker, K. R., *Introduction to Sequencing and Scheduling*, John Wiley & Sons, New York, 1974.
4. Ball, M. O. etc., *Network Models*, North-Holland, New York, 1995.
5. Frederickson, G. N., Hecht, M. S. and Kim, C. E., "Approximation algorithms for some routing problems," *SIAM J. Comput.*, Vol. 7, No. 2, pp. 178-193, 1978.
6. Gendreau, M., Hertz, A. and Laporte, G., "New insertion and postoptimization procedures for the traveling salesman problem," *Oper. Res.*, Vol. 40, pp. 1086-1094, 1992.
7. Garey, M. R. and Johnson, D. S., *Computers and Intractability : A Guide to the Theory of NP-Completeness*, W.H. Freeman and

Company, San Francisco, 1979.
8. Rubinovitz, J. and Wysk, R. A., "Task level off-line programming system for robotic arc welding-an overview," *Journal of Manufacturing Systems*, Vol. 7, No. 4, pp. 293-306, 1988.
9. Nemhauser, G. L. and Wolsey, L. A., *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, 1988.
10. Orloff, C. S., "A fundamental problem in vehicle routing," *Networks*, Vol. 4, No. 1, pp. 35-64, 1974.

1998년 3월 접수, 1999년 1월 채택