PCB CAD에서의 최적 배선을 위한 진화 프로그래밍을 이용한 자동 부품 배치

Evolutionary Programming-Based Autoplace for Optimal Routing in PCB CAD

한 웅 석*, 김 종 환*
Woong-seok Han*, Jong-Hwan Kim*

ABSTRACT

In this paper, a new method of finding a sub-optimal solution of an autoplacer which places electrical components automatically in PCB CAD tools. The software implementation of the proposed method can be viewed as a new type of floorplan based on evolutionary programming. To solve this problem, three kinds of operators and a fitness function are designed. Computer simulation results demonstrate the usefulness and effectiveness of the proposed scheme in the light of computation time and effort.

I. 서 론

PCB를 만들기 위하여 캐드(CAD)에서 작업을 할 때 부품간의 배선을 가능한 한 효율적으로 하기 위하여 부품의 위치를 바꾸고 부품의 방향을 바꾸어 보는 등 여러 작업 과정을 거쳐 부품 배치를 한다. 이런 작업은 많은 경험을 통해서 얻은 지식을 기반으로 수행되기 때문에 초보자가 하기에는 많은 어려움이 있다. 또한 숙련자라고 해도 쉽게 할 수 있는 것이 아니라 많은 노력과 시간을 요하게 된다. 따라서, PCB작업을 하면서 자동으로 부품의 배치를 최적으로 할 수 있는 방법을 찾기 위해 많은 연구가 되어 왔다.

현재에도 캐드 프로그램들은 이러한 자동 부품 배치를 제공한다. 하지만, 이러한 자동 부품 배치 기능이 캐드 프로그램에서 주된 기능이 아니기 때문에 좋은 성능을 제공하지 않고, 그러한 기능을 수행하는 방식 또한 좋은 알고리즘을 사용하지 않고, 단지 행렬을 이용한 규칙적인 배열을 이용하여 배치를 한다. 그래서, 다양한 부품 배치를 할 수 없고 부품이 배치되는 위치 또한 일정한 틀에 제한된다. 여러 논문에 이와 비슷한 것들이 논의되기도 하였는데, 이러한 접근 방식 중에 하나는 숙련된 기술자가 가지고 있는 지식을 바탕으로 전문가 시스템을 만드는 것이다 [1].

이러한 기존의 자동 부품 배치 프로그램의 단점을 개선하기 위해서는 다양한 경우를 발생시킬 필요가 있고, 부품이 배치되는 위치 또한 틀에 박혀 있으면 안된다. 그러기 위해서는 많은 경우를 생성시켜서 그것이 다른 경우보다 더 뛰어난 부품 배치인지를 검사하여야 한다. 이러면, 속도에서 굉장히 느려질 것이므로, 이러한 모든 경우를 따지지 않고도 효율적으로 최적 배치를 찾을 수 있는 새로운 기법이 필요하다.

*한국과학기술원 전기 및 전자공학과
Department of Electrical Engineering, KAIST

여기서 말하는 최적 배치란 배선을 할 때 최소 공간, 선간의 교차된 수, 선의 길이 등의 관점에서 최적으로 할 수 있도록 부품을 배치하는 것을 말한다.

본 논문에서는 진화 프로그래밍을 이용하여 전자 공학용 캐드중 하나인 CADSTAR 환경 하에서 동작하는 자동 부품 배치 방법을 제안하였다. 진화 프로그래밍(Evolutionary Programming) 기법은 개체들을 이용하여 확률적인 방법으로 좀 더 좋은 해를 찾아나가는 방법으로서 전역 최적해를 찾는데 효율적이라고 알려져 있다 [2][3]. 진화 프로그래밍은 처음에 여러 개의 개체를 임의로 초기화하여 그 다음에 그 부모를 바탕으로 확률적인 기법을 이용하여 자손을 생성하며, 부모와 자손 중 확률적인 적자생존을 거쳐서 살아남는 개체를 선택한다. 그리고, 그 살아남은 개체로부터 자손을 반복적으로 생산하여 최적의 자손을 골라낸다. 이를 실제 소프트웨어로 구현한 프로그램은 처음에 임의로 N_p개의 부품 배치 기판을 생성하고, 그 기판으로부터 확률적인 방법으로 그 자손을 하나씩 생성하여, 이러한 $2N_p$개의 기판 중에서 뛰어난 N_p개의 기판을 골라낸 후 다시 기판을 생성하는 방식을 사용하여 다시 검사하는 것을 반복한다. 그리하여, 최종적으로 성능이 가장 좋은 기판을 하나 선택하는데, 이것이 여기서 찾는 최적의 부품 배치이다. 본 논문에서는 10개의 부품이 제한 조건이기 때문에 8개의 부품으로 초기 결과를 보였으나 제안된 방법의 효율성을 보이기에는 충분했다. 이를 좀 더 확장하여 배선기와 직접적으로 결합하면 매우 좋은 결과를 얻을 것이다.

Ⅱ. 부품을 자동으로 배치하는 프로그램

이 절에서는 서론에서 제안한 부품을 자동으로 배치하는 자동 부품 배치기에 대해 설명하고자 한다. 이 프로그램의 목적은 가능한 한, 배선을 할 때 최소 공간, 선간의 교차된 수, 선의 길이 등의 관점에서 최적으로 할 수 있도록 부품을 배치하는 것이다.

2.1 자동 부품 배치

이 기능은 일반적으로 전자 공학용 캐드에서 PCB 작업을 하는 경우에 많이 사용한다. 이 기능은 자동 배선과 함께 PCB작업에서 굉장히 필요한 기능이다. 논리적 회로인 회로도를 만든 후 회로도에서 설계한 회로를 실제로 사용하는 PCB로 만들 때, PCB에 부품을 배치하는 것을 부품 배치라 한다. 부품 배치는 전기적으로 연결 되야 할 정보를 나타내는 연결선을 바탕으로 배치가 일어나게 된다. 배선은 이러한 연결선을 바탕으로 실제로 전기적으로 연결을 해주는 것을 말한다.

PCB작업에서 부품 배치는 부품 배치 다음 작업인 배선에 많은 영향을 준다. 시간과 노력이 많이 소모되는 PCB작업은 부품 배치에 따라서 배선의 방법이 여러 가지 형태로 바뀌기 때문이다. 때에 따라서는 PCB 기판을 만드는 비용까지 줄일 수 있게 된다. 이러한 부품 배치를 자동으로 해주는 것을 자동 부품 배치라 하고, 배선하기에 좋은 형태로 부품 배치해 주는 것이 자동 부품 배치기의 역할이 된다. 이 논문에서는 이러한 배선을 고려하여 자동 부품 배치를 하도록 하였다.

2.2 CDI 파일

CDI 파일이란 CADSTAR에서 사용하는 파일 형식으로, CADSTAR Data Input format의 약자이다. 여기에는 기판에 대한 가장 기본적인 정보들이 담겨져 있으며 아스키 형식으로 되어 있기 때문에, 사용자가 쉽게 알아볼 수 있다. 이 파일은 다음과 같은 정보를 가지고 있다.

1. PCB 기판의 이름
2. CADSTAR 기본 환경 설정에 대한 정보 (화면에 표시되는 색, PAD와 VIA의 모양과 크기, 회로선의 폭 등)
3. PCB에서 사용하는 부품에 대한 정보 (부품 라이브러리(library) 번호, PAD 위치와 모양 등)
4. 연결선에 대한 정보
5. 배선에 대한 정보

본 논문에서는 CADSTAR에서 만드는 CDI 파일을 읽어 들여서, 부품의 위치와 방향만 이용하여 자동 부품 배치하여 같은 형식의 CDI 파일로 저장을 하여 자동 부품 배치를 하게 된다.

Ⅲ. 진화 프로그래밍을 이용한 자동 부품 배치

3.1 진화 프로그래밍 (Evolutionary Programming)

진화 프로그래밍이란 진화라는 자연의 개념을 도입하여 최적의 개체(최적의 해)를 찾는 것이다. 이러한 최적의 개체를 찾기 위하여 진화 프로그래밍에서는 돌연변이 기법을 주로 이용한다. 이와 같은 돌연변이가 된 개체 중에서 환경에 잘 적응하는 것은 그 특성이 후손으로 전달되어 진화가 일어나게 된다. 진화 프로그램의 기본 흐름은 다음과 같다.

1. N_p개의 부모를 생성한다. 이것이 초기화인데, 부모가 좋으면 자식들이 좋은 것은 당연하다. 따라서 부모에 따라서 얼마나 많은 반복을 해야 하는 가가 결정이 된다. 그러나 일반적으로 임의로 부모를 생성한다.

2. 1에서 정한 N_p개의 부모로부터 N_o개의 자식을 생성한다. 자식을 생성할 때 어떻게 돌연변이를 만드는 가에 따라서 그룹의 환경 적응 속도가 바뀐다. 다양한 돌연변이는 국소해에 빠지는 것을 방지하기 때문에 돌연변이를 다양하게 생성하는 것 또한 중요한 일이다.

3. 1과 2에서 생성된 개체들이 환경에 얼마나 적응하는 가를 알아본다. 우리가 원하는 환경을 함수로 만들어서 1과 2에서 만든 개체들의 특성인 변수의 값을 함수에 넣어 함수값에 따라서 개체가 환경에 얼마나 적응을 잘하는가 알아본다. 여기서 함수의 값이 작은 쪽이 좋은가 또는 큰 쪽이 좋은가는 함수를 만드는 것에 따라 달라지게 된다.

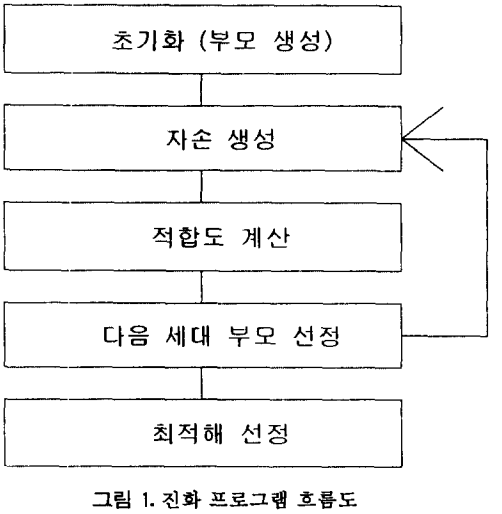

그림 1. 진화 프로그램 흐름도

4. 3에서 얻은 $N_P + N_O$개의 개체에 대한 환경 적응 정보를 이용해 환경에 잘 적응하는 개체를 N_p개를 선택해서 다음 세대의 부모로 선정을 한다. 그리고 이 여기서 구한 N_p개의 부모를 이용해서 2부터 다시 반복 수행을 하게 되는데, 반복 횟수는 앞에서 정하는 만큼 반복을 하게 된다.

5. 반복이 끝나면 최종적으로 살아남아 환경에 가장 잘 적응하는 개체가 우리가 원하는 해가 된다.

위의 방법을 도식화하면 그림 1과 같다.

진화 프로그래밍을 사용하기 위해서는 주어진 문제의 개체에 의한 유전인자형 표현과 이들의 초기화, 돌연변이 생성에 대한 개발, 그리고 원하는 최적에 대한 적절한 적합도 함수를 선정하여야만 좋은 결과를 얻을 수 있다.

3.2 자동 부품 배치

이 절에서는 진화 프로그래밍에 기초해서 부품을 자동 배치하는 알고리즘의 각 부분에 대하여 2절에서 설명한 것을 기반으로 하여 설명하고자 한다. 자동 부품 배치는 진화 프로그래밍에 적용시키기 어려우나, 각 개체를 간단화하고, 돌연변이 생성을 다양화하고, 적합도 함수의 적절한 표현으로 적용시킬 수 있었다.

3.2.1 기판 크기 설정

입력되는 각 부품의 가로 세로 길이의 최대값을 취해서 그 최대값의 합을 기판의 크기로 정한다. 이 기판 크기는 처음에 최대의 크기로 잡아 초기의 부품 배치에서 걸리는 시간을 줄이기 위함이다. 해를 구할 때는 기판 크기를 작게 하는 것도 포함이 되어 있으므로 이 크기는 중요하지 않다.

3.2.2 초기화

부품들을 제한된 크기의 기판 안에 배치한다. 일단 1세대 부모로 50개의 개체를 생성하여 이를 임의로 배치한다. 이에 기본적인 부품 배치에서 부품간에 겹치는 경우와 부품이 기판 밖으로 나가는 경우에는 다시 생성하게 하여 정상적인 부품 배치가 된 것만 골라낸다. 각 부품의 구조는 그림 2와 같이 유전인자형으로 표현할 수 있다.

여기서 x_i는 부품의 x위치를 나타내고, y_i는 부품의

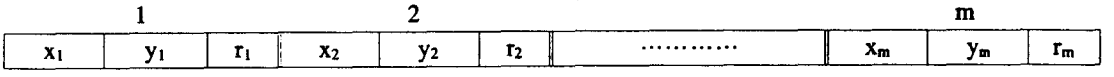

그림 2. 각 개체의 구조

y위치를 나타내고, r_i는 부품의 방향을 나타낸다. 그리고 m은 부품의 개수를 나타낸다.

3.2.3 자손 생성

여기서는 이미 생성된 직전 세대의 50개의 부모로부터 돌연변이 기법을 통해 각각 1개의 자손을 생성시켜서 전부 50개의 자손을 생성한다. 돌연변이 기법은 다음과 같은 순서로 진행된다.

1. 부품을 임의로 선택을 한다.

2. 돌연변이를 생성시킬 방법을 선택한다. 방법을 선택할 때는 숙련자의 경험을 바탕으로 만든 확률적 상황을 이용하여 임의로 선택한다. 실제의 PCB작업을 할 때 처음에는 부품의 위치를 서로 바꾸는 경우가 많고, 부품을 움직이는 범위도 넓다. 부품을 회전시키는 경우는 PCB작업을 하는 경우는 일정하게 일어난다. 그래서, 프로그램의 속도를 향상시키기 위해서는 처음에는 부품을 서로 바꿔 주는 확률을 크게 하고, 배치를 완료됨에 따라 확률을 감소시킨다. 부품이 이동하는 경우에도 배치가 완료됨에 따라서 범위가 작아지도록 해야 한다. 배치 상태에 대한 척도는 현재 기판의 상태를 나타내는 목적 함수 값이 될 수 있다. 하지만, 여기서 문제가 되는 것은 목적 함수 값을 부품의 수에 관계없이 정규화할 수 없기 때문에, 현재의 목적 함수 값이 기판의 어떤 상태를 나타내는지 알 수 없어 기판 상태에 따른 확률적인 선택이 불가능하다. 따라서, 이 논문에서는 세 가지 경우가 일어날 확률을 모두 같도록 하였다.

3. 2에서 정한 방법으로 1에서 정한 부품을 옮겼을 때, 부품들이 서로 겹치는지, 혹은 기판의 범위를 넘어가는지 판단한다. 만일 오류가 있을 경우에는 2로 가서 돌연변이를 생성하는 방법을 다시 선택한다. 그러나, 2로 가서 돌연변이를 생성하는 방법을 다시 택하는 과정을 10번 이상 반복할 경우에는 1로 가서 부품부터 다시 선택하여 오류가 없는 배치가 나올 때까지 반복하여 돌연변이를 만들어 낸다.

돌연변이를 만드는 방법에는 여러 가지가 있을 수가 있으나, 이 프로그램에서 사용한 방법들은 다음과 같다.

1. 부품을 회전한다. 여기서는 0, 90, 180, 270에 해당하는 방향만이 있기 때문에 이 4가지 경우 중에 1가지를 택해서 방향을 바꾸게 된다. 아래의 그림 3은 부품 회전의 경우를 나타내고 있다.

r_2'은 0, 90, 180, 270중에서 한가지 값을 갖는다. 단, 그러나 r_2와 같지 않다.

2. 부품을 이동시킨다. 부품을 이동시키는 방향은 +x, −x, +y, −y의 4가지 방향이고 이 방향은 임의로 결정을 하게 된다. 움직이는 양도 임의로 결정하게 된다. 아래의 그림 4는 y방향으로 부품이 이동하는 경우를 나타낸다.

y_2'는 $y_2+\alpha$가 된다. 여기서 α는 실수이며 임의로 결정된다.

3. 두개의 부품의 위치를 서로 바꾼다. 두개의 부품은 임의로 결정한 후에 두개의 x 위치, y 위치, 방향을 서로 바꾸게 된다. 아래의 그림 5는 두 부품을 바꾸는

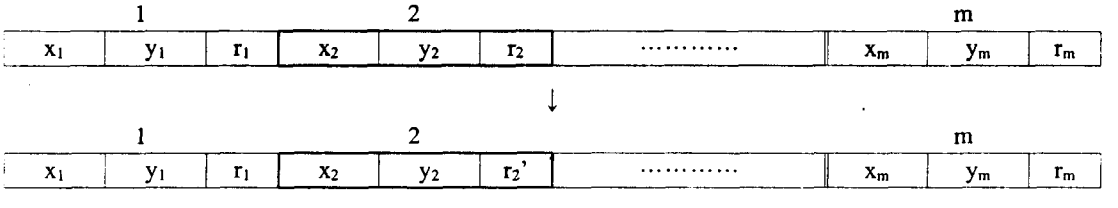

그림 3. 부품의 회전

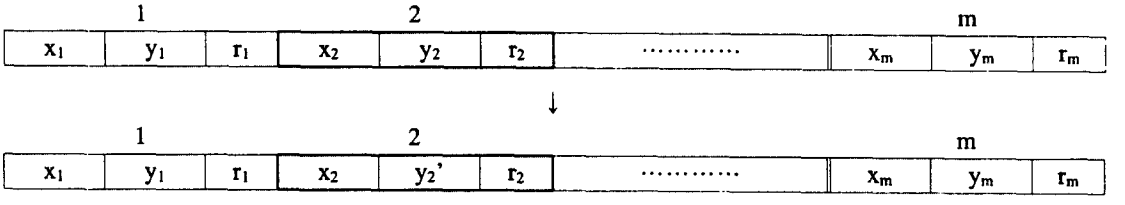

그림 4. y방향으로 부품을 이동

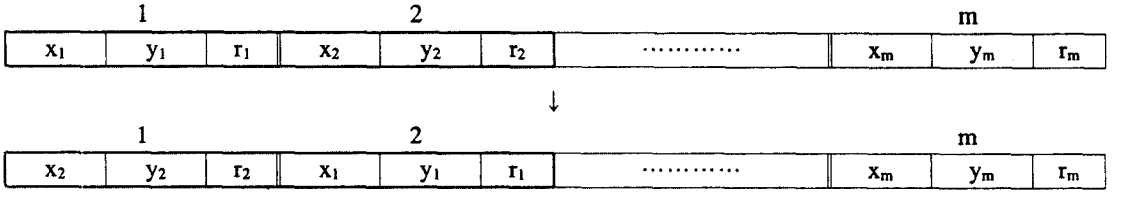

그림 5. 2개 부품 교환

경우를 나타내고 있다.

3.2.4 적합도 계산

이렇게 생성된 부모와 자손의 자료를 가지고 목적 함수에 적용시켜 현재의 기판이 어느 정도나 조건을 만족시켰는지 결정한다.

여기서 사용한 목적 함수 f는 다음과 같다.

$$f = \frac{U+1}{\left\{\sum_{ij}^{Q_{ij}} L_{ijk}\right\} \times (C+1) \times S}$$

위의 식에서 사용된 변수는 다음과 같다.

U는 부품들간에 꼬이지 않은 연결선의 개수를 나타낸다.

Q_{ij}는 부품 i와 j사이의 연결선의 개수를 나타낸다. (단, $1 \le i \le m$, $1 \le j \le m$이고, $i \ne j$)

L_{ijk}은 부품 i, j의 연결선 중에서 연결선 k의 최단거리를 나타낸다. (단, $1 \le i \le m$, $1 \le j \le m$, $1 \le k \le n$이고, $i \ne j$)

m은 기판에서 사용된 부품의 수를 나타낸다.

n은 두 부품 사이에서 서로 연결된 선의 수를 나타낸다.

C는 부품들간에 연결선이 꼬인 것의 개수를 나타낸다.

S는 현재 배치된 부품에 의해 만들 수 있는 기판의 최소 크기를 나타낸다.

위의 목적 함수는 선의 길이를 최소화하고, 연결선들이 꼬이지 않고, 기판의 크기를 줄일 수 있도록 만들었다. U와 C에 1을 더하는 이유는 꼬인 선이나 꼬이지 않은 선이 없을 경우에 목적 함수의 값이 0이 나오거나 무한대가 나오는 것을 방지하기 위해서이다. 선의 길이를 최소화하는 것은 부품들 사이에 밀집도를 높이는 것이고, 연결선들이 꼬이지 않게 하는 것은 배선할 때에 선이 꼬이지 않도록 하기 위해서이다. 여기서 부품들 사이에 밀집도가 높아지면 기판의 크기가 줄어드는 것은 당연하지만, 밀집된 모양이 항상 직사각형이 되지 않는다. 따라서 기판의 크기에 대한 것을 넣어서 부품 배치의 모양이 직사각형에 가깝도록 하였다.

위에 정의된 목적 함수 f는 값이 가장 큰 기판이 가장 좋은 배치를 가진다고 할 수 있다.

3.2.5 승수의 계산

100개의 개체가 각각 자신 외에 10개의 개체를 난수적으로 선택을 하여, 위에서 구한 목적 함수 값을 기준으로 비교를 하여 좋은 개체에 승수를 하나씩 더한다.

3.2.6 선택

3.2.5에서 구한 승수의 개수를 가지고 승수의 수가 많은 순서로 정열을 한다. 다음 세대의 부모로 위에서부터 50개의 개체를 선택하고 다시 자손을 만들기 위해 3.2.3부터 다시 반복한다. 그리고 이러한 반복은 앞에서 정한만큼 일정 회수를 반복한다.

3.2.7 최적해

3.2.3 부터 3.2.6의 과정을 일정 회수 반복한 후 가장 목적 함수의 값이 큰 것을 최적의 기판으로 간주한다.

Ⅳ. 시뮬레이션 및 검토

이 프로그램에서 부품 수는 10개로 제한하였다. 그리고, 시뮬레이션에 사용한 부품의 개수는 4개와 8개이다. 이 때 세대를 발생하는 반복을 각 400번, 1000번을 한 후 그 때 나오는 기판의 결과를 보았다. 이 경우에는 부품의 개수에 비례하지 않고 연결된 선수에 비례함을 관찰할 수 있다. (시뮬레이션에 대한 결과는 그림 6, 그림 7, 그림 8, 그림 9를 참조)

그리고, 이 프로그램을 사용할 때 주의 할 점은 이 프로그램이 DOS환경 하에서 만들어 졌기 때문에 CADSTAR에서 부품을 그린 후, DOS에서 부품 배치를 해준 후에 다시 CADSTAR에서 결과로 나온 파일을 읽어 들이면 부품이 배치된 결과를 볼 수 있다.

이 프로그램을 이용해서 나온 결과들 중에서 부품이 4개인 경우를 보면 (그림 6, 그림 7 참조), 그림 상으로 약간의 변화를 볼 수 있다. 두 그림에서 기판의 크기나 꼬인 선의 개수에는 변화가 없으나 전체 선의

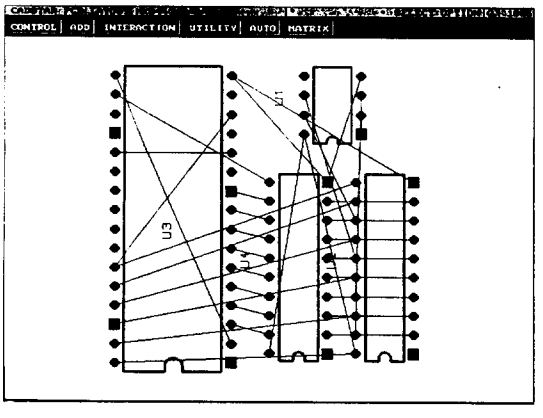

그림 6. 부품 4개, 반복이 400인 경우

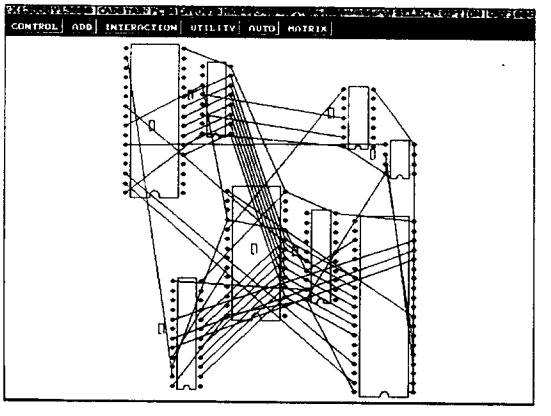

그림 8. 부품 8개, 반복이 400인 경우

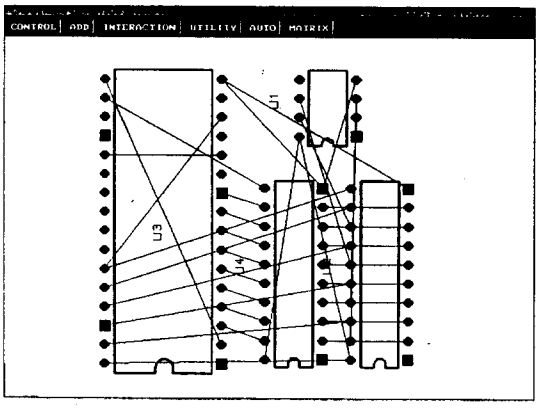

그림 7. 부품 4개, 반복이 1000인 경우

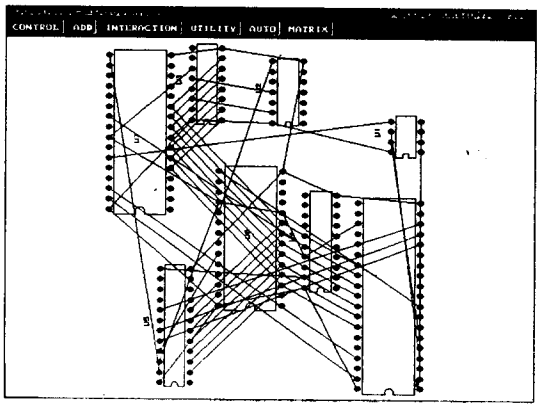

그림 9. 부품 8개, 반복이 1000인 경우

길이가 그림 7에서 더 짧아졌다. 이 경우에는 부품의 수가 적고 연결선이 꼬이지 않은 것이 많으므로 적은 수의 반복으로 준 최적해에 도달하였다. 이 시뮬레이션은 어느 정도 최적화가 되면 반복의 수를 늘려도 변화가 없음을 보여준다.

부품이 8개인 경우를 보면 (그림 8, 그림 9 참조), 변화가 있음을 뚜렷하게 알 수 있다. 그림 8보다 그림 9가 확대되어 있는 그림이기 때문에 400번일 때 보다 1000번일 때 부품들이 더 밀집이 되어 있는 것을 볼 수 있고, 연결선도 꼬여 있는 선의 개수가 줄어든 것을 볼 수 있다. 부품이 8개인 경우의 연결선이 복잡해서 꼬인 선이 많이 보이기는 해도 배선할 때 어느 정도 꼬인 것은 풀어 나가면서 배선을 할 수 있다. 그러나 너무 많이 꼬여 있는 경우에는 배선에서도 해결을 할 수가 없기 때문에 이러한 문제점을 부품 배치에서 해결하는 것은 중요하다. 일단 여기서는 연결선이 부품이 4개일 때보다 더 복잡하기 때문에 준 최적해에 쉽게 도달하지 않음을 알 수 있으나 세대수가 지남에 따라 진화 프로그래밍이 준 최적해를 찾아가고 있다는 것을 보여준다.

여기에서 사용한 목적 함수는 일반적으로 다른 곳에 사용하는 진화 프로그래밍과 다른 형태를 지니고 있다. 일반적으로 목적 함수는 각 개체의 특성을 나타내는 변수들에 상수를 곱해서 서로 더해서 구하는 것이 보통이다. 이 경우 사용되는 상수들은 서로의 값들이 변하는 범위나 자리 수를 바꾸어서, 개체 특성에 가중치를 주는 것이다. 그러나 이 프로그램에서 사용된 목적 함수를 보면 서로 더하는 것이 아니라 서로 곱하여 목적 함수를 구했다. 여기서 목적 함수를 곱하기로 한 이유는 반복이 일어날 때마다 개체에 대한 특성의 값이 너무 많이 변하여, 가중치를 나타내는 상수만으로 해를 구할 수 없었기 때문이다. 서로 값을 곱하게 되면 어느 특성 값이라도 목적 함수가 영향을 받게 된다. 그러나 여기에도 문제점은 존재한다. 서로 값을 곱하게 되면 각 특성 값들에 가중치를 넣을 수 없다는 것이다. 그러나 이러한 점에도 불구하고 여기서 사용한 곱셈의 방법이 일반적으로 사용하는 가중치 방법보다 해를 더 빠르게 찾았다.

앞에서도 말했듯이 목적 함수 값에 따라서 돌연변이를 생성하는 방법이 바뀌어야 한다. 그러기 위해서 먼저 목적 함수를 부품의 개수와 연결선의 개수에 관계없이 정규화를 해야 한다. 그러면 목적 함수에 따라 돌연변이를 만드는 방법을 택하는 확률을 목적 함수에 따라 다르게 하여 최적화를 빠르게 찾도록 할 수 있다.

여기서는 부품들간에 연관성을 생각하지 않고 부품 배치를 하였다. 그러나 부품 배치를 빠르게 하기 위해서는 부품들간에 연관성을 생각하여 연관이 많은 것을 모아 부품 배치하는 것도 하나의 해결 방법이 될 수 있다.

실제로 수많은 경우의 수중에 어느 것이 더 좋은 최적화인지 구별해 내기 어렵다. 하지만 기판이 최적화가 되어 있는지, 최적화를 찾아가는 지를 알아볼 수 있다. 그리고 이 프로그램은 하나의 준 최적해를 찾아내고, 준 최적해에 접근하는 것을 볼 수 있다.

본 논문에서는 돌연변이 발생 결과에서 부품들이 서로 겹치거나, 혹은 기판의 범위를 넘어서는 오류가 발생한 경우에는 이를 버리고 다시 돌연변이를 발생시키는 방법을 택하였다. 하지만, 이와 같은 문제는 이상(two-phase) 진화 프로그래밍 기법[9]을 이용하여 오류를 수정한다면 더 좋은 결과를 얻을 수 있을 것이다.

V. 결 론

본 논문에서는 진화 프로그래밍을 이용하여 배선을 고려한 자동 부품 배치를 제안하였다. 시뮬레이션 결과, 제안된 방법은 하나의 준 최적해에 접근해 가는 것을 보여준다.

여기서는 자동 부품 배치에 대한 최적화 기법을 논하였다. 부품 배치만으로 PCB 기판 전체에 대한 최적화를 할 수 없다. 따라서, 자동 부품 배치는 자동 배선기와 직접적으로 연관이 되어 있기 때문에 부품 배치를 하면서 배선을 동시에 하는 프로그램을 만드는 것이 이상적이다. 이런 형태로 자동 부품 배치기와 자동 배선기를 만들어야 부품 배치만 하는 프로그램과 배선만 하는 프로그램의 한계를 넘어서 더 좋은 최적해를 구할 수 있을 것이며, 이는 추후의 연구 과제로 연구 중에 있다.

참 고 문 헌

1. Yoshimura, H. et al.: Knowledge-based placement and routing system for printed circuit board. Proc. PRCAI'90, pp. 116-121, 1990
2. Bäck, T. and Schwefel, H. -P. "Evolutionary Computation: An Overview," in Proc. of IEEE int'l Conf. on Evolutionary Computation, pp. 20-29, 1996
3. D. B. Fogel *Evolutionary Computation*: Toward a New Philosophy of Machine Intelligence, IEEE Press, Piscataway, NJ, 1995
4. Bäck, T., Hoffmeister, F., and Schwefel, H. -P.: A Survey of Evolution Strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 2-9, Morgan Kaufmann Publishers, 1991
5. Bäck, T. and Schwelfel, H. -P., Evolutionary Programming and Evolution Strategies: Similarities and Differences. In Fogel, D. B. and Atmar, W., editors, *Proc. of the Second Annual Conference on Evolutionary Programming*, pp. 11-22, La Jolla, CA. Evolutionary Programming Society, 1993
6. Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wiley, 1989
7. Michalewicz, Z. *Genetic Algorithms + Data Structure = Evolution Programs*. Springer, 1992
8. Saravanan, N. and Fogel D. B., Evolving Neurocontrollers using Evolutionary Programming. In *Proc. of the WCCI '94, Orlando, Florida*, pages 217-222, 1994
9. J. -H. Kim and H. Myung, "A two-phase evolutionary programming for general constrained optimization problem," in *Proc. of the Sixth Annual Conference on Evolutionary Programming*, MIT Press, Cambridge, MA, 1996, in press.

한 웅 석(Woong-seok Han) 정회원
한국과학기술원 전기 및 전자공학과 학사과정에 재학중. 제 1회 KAIST 마이크로 로봇 축구 대회 입상. 주요 관심분야는 지능제어 및 로보틱스, 진화 연산을 이용한 최적화, 회로 설계 및 CAD.

김 종 환(Jong-Hwan Kim) 정회원
1981년: 서울대 공대 전자공학과 졸업.
1983년: 동 대학원 전자공학과 졸업(석사).
1987년: 동 대학원 전자공학과 졸업(공박).
1992년 8월~1993년 8월: 미국 Purdue 대학 교환교수.
현재: 한국과학기술원 전기 및 전자 공학과 부교수.
1988년: 제3회 춘강 학술상 수상. IEEE Transactions on Evolutionary Computation 및 International Journal of Intelligent & Fuzzy Systems의 Associate Editor, IEEE ICEC '95, '96, '97, IEEE IECON '96, IEEE SMC '96, SEAL '96, ICGA '97, EP '97, ICCIMA '97, ICKES '97, ICRM '97의 Int'l Program Committee Member, ICRA '97의 진화 연산 위원회 위원장 겸 Int'l Program Committee Member, 마이크로 로봇 월드컵 축구대회(MIROSOT) 조직 위원장. 주전공 또는 관심분야는 진화 마이크로 로봇 공학.