Development of a Heuristic Method for Solving a Class of Nonlinear Integer Programs with Application to Redundancy Optimization for the System using Multi – Processor

Jae – Hwan Kim, Jang – Wook Kim

Department of Applied Mathematics, Korea Maritime University, Pusan, Korea

Abstract

This study is concerned with developing a heuristic algorithm for solving a class of nonlinear integer programs(NLIP). Exact algorithms for solving a NLIP either may not exist, or may take an unrealistically large amount of computing time. This study develops a new heuristic, the Excursion Algorithm(EA), for solving a class of NLIP's. It turns out that excursions over a bounded feasible and/or infeasible regio is effective in alleviation the risks of being trapped at a local optimum.

The developed EA is applied to the redundancy optimization problems for improving the system safety, and is compared with other existing heuristic methods. We also include simulated annealing(SA) method in the comparision experiment due to its popularity for solving complex combinatorial problems. computational results indicate that the proposed EA performs consistently better than the other in terms of solution quality, with moderate increase in computing time. Therefore, the proposed EA is believed to be an attractive alternative to other heuristic methods.

1. 서 론

본 연구에서 다루고자 하는 시스템의 신뢰도 중복설계는 선박, 비행기, 통신 시스템등의 안전을 위해 자원이나 기술적 요인에 따른 다양한 제약하에서 하부시스템(subsystem)의 중복(redundancy)설계를 통하여 시스템의 신뢰도를 최적화하는 문제이다. 이 문제는 일종의 비선형 정수계획문제로 분류된다[4]. 비선형 정수계획의 해법은 크게 정확한 방법(exact method)과 발견적 해법(heuristic method)으로 구성되는데, 정확한 방법([10], [15], [17])은 global 최적해를 구할 수 있는 장점이 있지만 중복설계문제를 포함한 모든 비선형 정수계획문제에 대해 개발된 것은 아니며, 개발된 방법조차도 계산상의 복잡도(computational complexity)에 의한 계산노력이 너무 많이 소요되기 때문에 실용화하기 어려운 단점이 있다(Glover [5]). 이를 극복하기 위해 다양한 발견적 방법들(heuristic algorithms)이 개발되었는데([6], [7], [8], [14]), 이 방법들은 정확한 방법에 비해 계산시간은 적게 소

요되지만 국부최적해(local optimum)에 도달할 수 있는 단점이 있다.

이러한 발견적 방법들은 축차적개선(sequential improvement[3])과정을 바탕으로 확장 및 보완된 것이다. 즉, 초기해로 부터 exchange 방법에 의하여 근접해 있는 영역을 탐색해서 목적함수를 개선시켜 나가는 것이다. 그러나, 이러한 방법은 global 최적해와 거리가 있는 국부 최적해에 도달할 위험이 있다. 따라서, 이러한 위험을 줄이고자 여러가지 전략이 고려되고 있는 데, 예를 들면 pairwise exchange방법을 확장하여 2 - way 이상의 exchange([3], [11])를 수행하는 방법과 oscillating 절차[6] 등이 있다.

발견적 방법에서 국부 최적해에 빠지는 위험을 줄이고자 개발된 현재까지의 전략들은 그 축차적 개선 과정을 기능해의 영역(feasible region)에 국한하고 있다는 것이 특징이다. 그러나, 질이 높은 해를 얻기 위해 탐색 영역을 가능해 뿐만 아니라 비가능해의 영역까지 확장하는 전략을 생각해 볼 수 있는데, 이러한 전략을 택하고 있는 발견적 방법은 Mitchell[14]에 의해서 연구되었다. Mitchell의 소위 "DETMAX Algorithm"은 최적실험설계를 위한 방법으로서 통계권에는 잘 알려져 있으나 OR권에는 거의 알려져 있지 않은 것으로 여겨진다. DETMAX Algorithm의 한 가지 단점은 제약식이 있는 문제를 다루기 어렵다는 점이다. 따라서, 비가능해의 영역까지 탐색을 확장한다는 DETMAX의 전략을 살리되, 제약식이 있는 비선형 정수계획 문제를 다룰 수 있는 발견적 해법의 개발이 요구되고 있다.

그러므로, 본 연구에서는 위의 문제에 대해 국부최적해에 도달하는 위험을 줄이고자 기존의 축차개선과정의 exchange방법을 개선하여 기능해의 영역과 비가능해의 영역에서 탐색을 수행하는 EA(Excursion Algorithm)라는 새로운 발견적 방법을 개발하여 정확한 방법이 아직 개발되지 않은 문제에 대한 유용한 해법으로 제시하고자 하며, 아울러, 본 연구에서 개발한 새로운 방법인 EA를 시스템의 안전을 위한 신뢰도 중복설계 문제에 적용하였다. 또한 EA의 성능을 평가하기 위해, 적정한 규모의 중복설계문제를 빠르게 처리할 수 있는 다중프로세서(multi - processor)의 전용 시스템을 설계하여 기존의 발견적 방법들([9], [18])과 최근 인공지능분야에서 유용한 대안으로 등장하고 있는 Simulate Annealing(SA) 알고리즘([2], [13])과 그성능을 비교하였다.

1. 1 기호

비선형 정수계획에 속하는 중복설계문제에 적용한 새로운 발견적 해법인 EA를 설명하기 위한 기호는 다음과 같다.

- n : 하부시스템의 수.
- m : 제약식의 수.
- x_i : 번째 하부시스템의 부품의 갯수. 양의 정수, $i=1, 2, \cdots, n$.
- x : $(x_1, x_2, \cdots, x_n)$.
- x_0 : 초기 가능해.
- x^l : excursion 탐색 중 어느 시점까지의 최적해.
- x^* : 알고리즘 수행후에 바라견된 최적해.
- x^c : excursion 탐색 중 어느 시점까지의 최적해.
- $g_{ji}(x_i)$: i번째 하부시스템에 소요되는 j번째 자원의 양.
- b_j : j번째 자원의 최대 사용 가능량.
- b_j^c : $b_j - \sum_{i=1}^{n} g_{ji}(x_i)$.
- r_i : i번째 하부시스템의 부품의 신뢰도
- $R_i(x_i)$: x_i개의 부품을 중복설계 했을 때 i번째 하부시스템의 신뢰도.
- $Q_i(x_i)$: x_i개의 부품을 중복설계 했을 때 i번째 하부시스템의 비신뢰도.
- $R_s(x)$: 전체 시스템의 신뢰도.
- $x(\pm i)$: i번째 하부시스템의 부품을 하나 증가시키거나(+) 감소시킴(-).
- $\Delta g_{ji}(\pm i)$: i번째 하부시스템의 부품을 하나 증가시키거나(+) 감소시켰을 때(-)
- $\Delta R_s(\pm i)$: i번째 하부시스템의 부품을 하나 증가시키거나(+) 감소시켰을 때(-) 전체 시스템 신뢰도의 변화량.
- F- 집합 : excursion 탐색이 실패했을 때 탐과정에서 발생한 모든 해들의 집합.
- E- 집합 : excursion 탐색 동안에 발생한 해들의

집합.

FR : 가능해의 영역(feasible region).

BFR : 한정된 가능해의 영역(bounded feasible region).

BIFR : 한정된 비가능해의 영역(bounded infeasible region).

Δ_j : BFR 또는 BIFR의 경계를 결정해주는 상수.

1.2 가정

1. 시스템과 하부시스템은 모두 coherent하다.
2. 시스템은 n개의 하부시스템으로 구성되고, 각각은(1 - out - of - x_i : G)이다.
3. 모든 부품 상태들은 통계적으로 독립이다.
4. i번째 하부시스템에 소요되는 j번째 자원의 양은 x_i의 증가 함수 이며, 각 하부시스템의 j번째 자원 소비량은 j번째 자원의 전체 소비량에 대해 가법성을 갖는다.
5. 전체 시스템 신뢰도 R_s는 R_i이나 Q_i의 함수로 주어진다.

1.3 모형

본 연구에서 다루고자 하는 시스템의 안전을 위한 중복설계 문제는 다음과 같은 비선형 정수계획 모형이다.

(P1) maximize $R_s(x)$

subject to

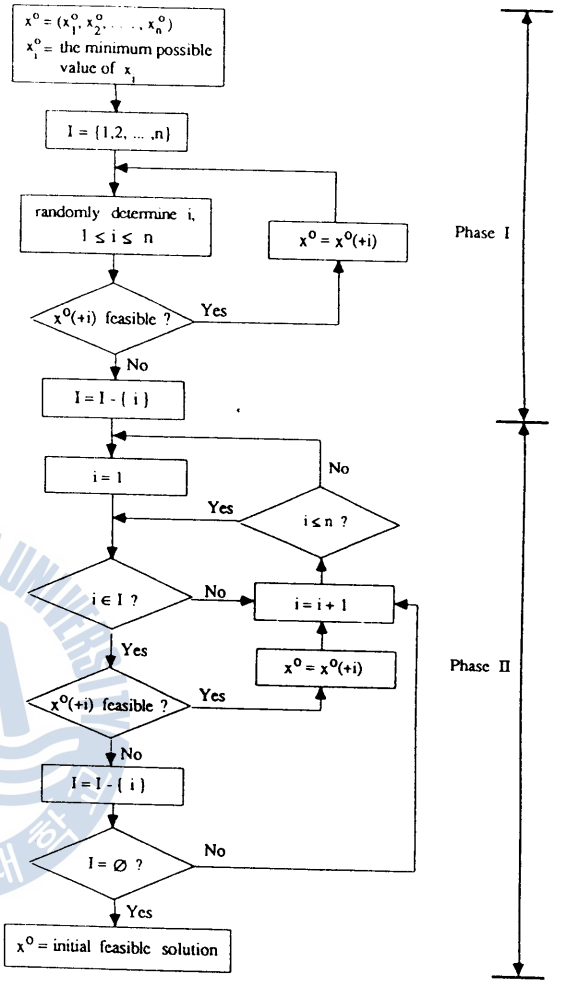
$\sum_{i=1}^{n} g_{ji}(x_i) \le b_j, j=1, 2, \cdots, m$

$x_i \ge 1$, 정수, $i=1, 2, \cdots, n$.

2. 발견적 해법의 개요

2.1 초기해의 구성

중복설계 문제에서 *EA*의 초기해를 구성하기 위하여 두가지 방법을 작용하였다. 3.1절의 예제에서는 <그림 1>의 two - phase 방법에 의하여 랜덤하게 구성한 초기해를 사용하였고, 4장의 전산실험에서는 기존의 다른 발견적 방법과의 성능을 비교하기 위하여 Kohda/Inoue[9]에서 언급된 extended Nakagawa/Nakashimal[16]방법에 의해 초기해를 구성하였다.



<그림 1> 초기해의 구성을 위한 **two - phase** 방법

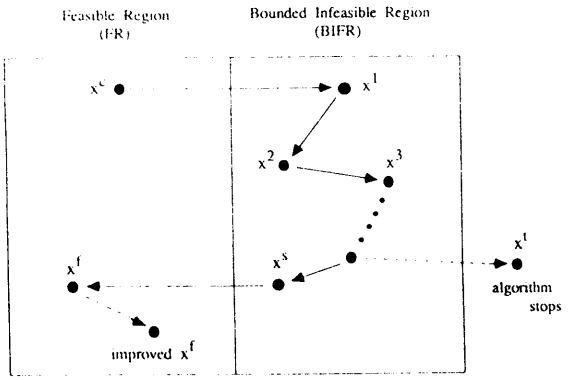
2.2 *EA*의 F - 집합에 의한 탐색

*EA*은 <그림 2>에서와 같이 현재의 최적해 x^c를 시발점으로 하여 BIFR을 향해 "excursion"을 시작한다. 이 excursion은 다음과 같은 두 경우 중 어느 하나로 끝나게 된다.

1) FR의 x^f로 되돌아 오는 경우.

2) BIFR을 벗어나 x^c로 끝나는 경우.

경우 2)일 때 *EA*은 중단되고, 그때의 최적해 x^*는 x^c가 된다. 한편, 경우 1)일 때는 <그림 3>의 절차에 의해 제약식의 여유분(slack)을 채워 xf를 개



<그림 2> *EA*의 탐색절차

선하여 현재의 최적해인 x^c와 비교하게 된다. 만약, 목적함수 $R_s(x^f)>R_s(x^c)$이면, 개선된 해가 찾아진 경우이므로 현재의 최적해 x^c를 x^f로 대체하고 새로운 excursion을 시작한다. 이와는 반대로, $R_s(x^f)\leq R_s(x^c)$이면, excursion 탐색이 실패 했으므로 excursion 동안에 탐색된 해들(<그림 2>에서 $x^1, x^2, \cdots, x^s$)을 F-집합에 저장시키고, 다시 현재의 최적해인 x^c를 시발점으로 하여 개선된 가능해를 찾기 위한 탐색을 시작한다.

F-집합에 의한 excursion 탐색은 <그림 2>에서 *BIFR*에 있는 한 점 x^t에 대해 다음과 같은 규칙을 적용하여 진행된다.

규칙1) $x^t\in F$이면, $x^{t+1}=x^t(+i)$ (forward move)

규칙2) $x^t\in F$이면, $x^{t+1}=x^t(-i)$ (backward move) (1)

규칙 2)는 x^t가 F-집합에 없으므로, 지금가지 고려되지 않은 새로운 해가 찾아진 경우이다. 따라서, 개선된 가능해를 생성할 가능성이 있다고 보고 제약식을 만족시키기 위해서 *FR*로 되돌아 오게 하는 것이다(backward move). 규칙 1)은 x^t가 F-집합에 있으므로, 개선된 가능해를 생성하는데 실패했던 해이다. 그러므로, *FR*로 되돌아 올것이 아니라 좀 더 좋은 해를 찾기 위해 *BIFR*에 있는 새로운 해를 탐색하게 하는 것이다(forward move). 이러한 BIFR에서의 excursion 탐색을 통하여 국부최적해에 도달하는 위험을 줄이고잔 하는 것이 *EA*의 기본 구상이다.

2.3 중복을 늘이거나 줄이는 기준

Excursion 탐색 중 비가능해인 x^t가 F-집합에 있으며, 다음과 같은 "forward move" 선정 기준에 의하여 해당되는 하부시스템의 중복을 늘리도록 하였다.

$$\max_{1\leq i\leq n}[\Delta R_s(+i)/\sum_{j=1}^{m}(\Delta g_{ji}(+i)/b_i)] \quad (2)$$

윗 식은 중복을 하나 늘림으로써 야기되는 신뢰도 증가량의 비용증가분에 대한 상대적 비율이 가장 큰 하부시스템을 선정하고자 하는 것이다.

이와는 반대로, 중복을 하나 줄이고자 할 때, 어떤 하부시스템을 선정 할 것인가 하는 "backward move"선정 기준은 다음과 같이 주어진다.

$$\max_{1\leq i\leq n}[\Delta R_s(+i)/\sum_{j=1}^{m}(\Delta g_{ji}(-i)/b_i)] \quad (3)$$

식(2)의 기준은 Shi[18]가 제시한 다음과 같은 기준으로 대치될 수도 있다.

$$\max_{1\leq i\leq n}[\Delta R_s(+i)/\prod_{j=1}^{m}(\Delta g_{ji}(+i)/b_i)] \quad (4)$$

2.4 EA의 중단 규칙

Exeursion 탐색이 진행되면서 x^c가 계속 개선되면 목적함수의 향상이 점점 어려워지고 F-집합은 확대된다. 다라서, excursion 도중의 해 xf또는 x^t는 규칙 1)에 의해 현재의 최적해 x^c로 부터 점점 더 멀어지게 되어 궁극적으로 *BFR* 또는 *BIFR*을 넘어서게 되며 이때 EA는 중단된다. *BFR*과 *BIFR*은 (P1)의 제약식을 이용하여 다음과 같이 정의할 수 있다.

BFR :

$$b_j-\sum_{i=1}^{n}g_{ji}(x_i)\leq\Delta_j,\ j=1,2,\cdots,m$$

BIFR :

$$\sum_{i=1}^{n}g_{ji}(x_i)-b_j\leq\Delta_j,\ j=1,2,\cdots,m \quad (5)$$

여기서, Δ_j는 양의 상수로서 사용자에 의해 정해지는 값이다. *EA*은 *BIFR*의 경계를 넘어서면 중단된다. 따라서, $b_j^c=b_j-\sum_{i=1}^{n}g_{ji}(x_i)$라고 정의하면, *EA*의 중단규칙은 다음과 같다.

$EA : |b_j^c| > \Delta_j,\ j=1, 2, \cdots, m$ (6)

Δ_j는 알고리즘 수행에 앞서 결정해 주어야 하는 상수인데, Δ_j를 크게 설정하면 BFR과 BIFR이 넓어지게 되어 개선된 해를 찾을 가능성이 증가하는 반면, 알고리즘의 수행시간이 길어 지는 단점이 있다. 한편, Δ_j를 작게 설정하면, 이와는 반대로 알고리즘의 수행시간은 줄어들지만, BFR과 BIFR이 좁아져 개선된 해를 찾을 가능성이 적어지는 단점이 있다. 따라서, Δ_j는 문제의 성격에 따라 적절히 선택되어야 한다.

Δ_j를 결정하는 방법에는 여러가지가 있을 수 있으나, 본 연구에서는 2가지 방법을 제시하고자 한다. 첫번째 일반적인 방법은 다음과 같은 식에 의해 Δ_j를 결정하는 것이다.

$\Delta_j = \alpha_i b_i$ (단, α_j는 양의 상수).

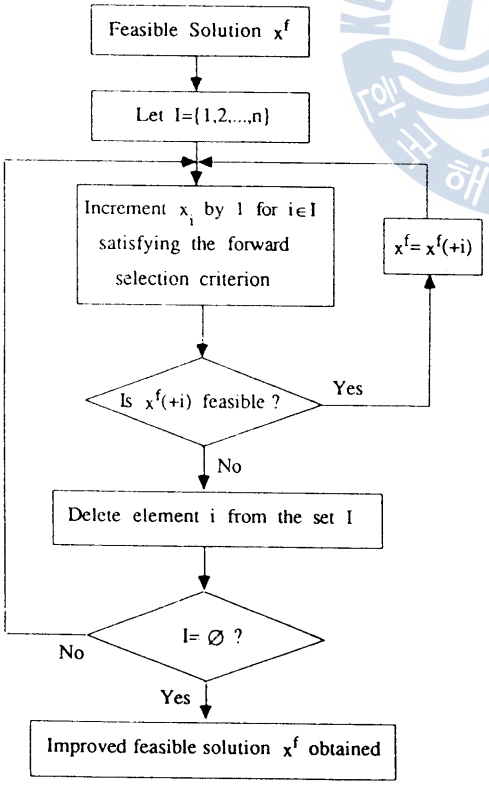
다른 방법은 몇 개의 결정변수가 한 단위 증가할 수 있는 여유분을 Δ_j로 결정하는 것이다. 예를 들어, 제약식이 선형인 경우, 즉 $g_{ji}(x_i)=c_{ji}x_i$일 때, Δ_j를 다음과 같이 설정해 줄 수 있다.

$\Delta_j = 2 \times c_{j(1)},\ j=1, 2, \cdots, m$

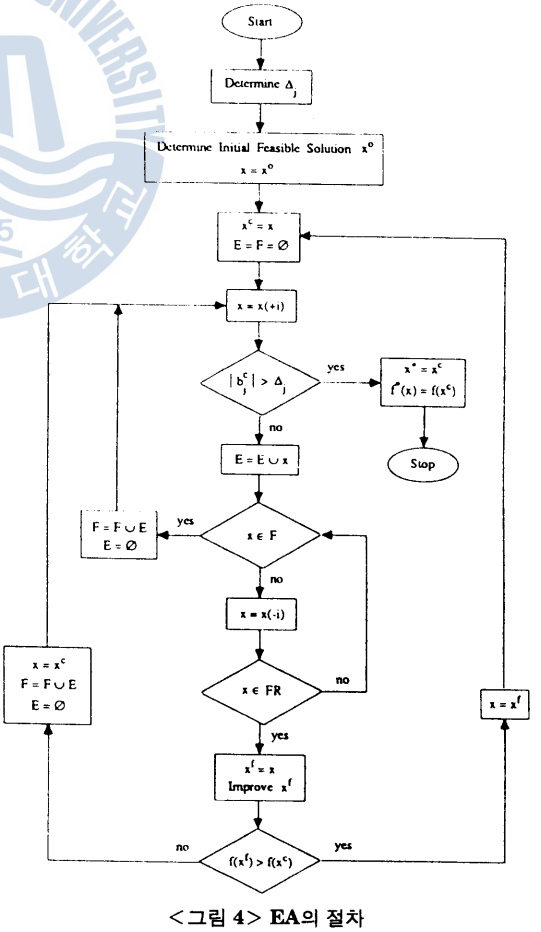
여기서, $c_{j(1)}$은 $c_{ji}(i=1, 2, \cdots, n)$ 중에서 가장 큰 값이다.

2.5 x^f의 개선

*BIFR*에서의 탐색중에 목적함수를 증가시킬 가능성이 있는 새로운 해가 찾아지면 *FR*로 되돌아오게 하는데, 일단 *FR*로 되돌아 온 가능해인 x^f에 대하여 <그림 3>에서와 같이 여유분(slack)을 채움으로써 목적함수를 개선시킨 다음, 현재의 최적해와 비교하도록 하였다.



<그림 3> x^f의 개선



<그림 4> EA의 절차

2.6 *EA*의 구체적인 단계

본 연구에서 개발한 EA의 단계는 다음과 같다 (<그림 4> 참조).

0. Δ_j를 결정한다.
1. 초기해 x^0를 구성한다. $x=x^0$.
2. $x_c=x$. $E=F=\phi$
3. $x=x(+i)$.
   $|b_i^c|>\Delta_j$이면, 단계 7로 간다.
4. $E=E\cup x$.
   $x\in F$이면, $F=F\cup E$, $E=\phi$. 단계3으로 간다.
   $x\notin F$이면, x=$(-i)$
5. $x\notin FR$이면, 단계 4로 간다.
   $x\in FR$이면, $x^f=x$. x^f를 개선시킨다.
6. $R_s(x^f)>R_s(x^c)$이면, $x=x^f$. 단계 2로 간다.
   $R_s(x^f)\le R_s(x^c)$이면, $x=x^c$.
   $F=F\cup E$, $E=\phi$. 단계 3으로 간다.
7. 중단, $x^*=x^c$, $R_s^*(x)=R_s(x^c)$.

위의 단계4에 의해 BIFR에서 탐색 중에 있던 해가 F- 집합에 있으면 그 이전까지의 탐색 중에 있던 해들은 F- 집합에 입력되고 forward move를 진행하므로 cycling은 일어나지 않는다.

3. 적용 예

*EA*의 탐색전략에 대한 이해를 돕기 위해, 다음 2개의 예제에 대하여 *BIFR*에서만 탐색을 수행하는 *EA*을 적용한 결과는 다음과 같다.

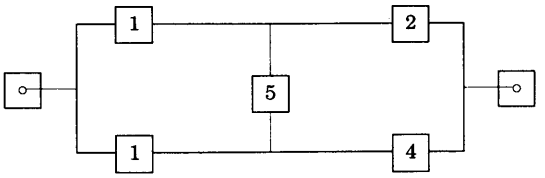
3.1 예제 1

Aggarwal[1], Kohda/Inoue[9], Shi[18]에서 고려한 <그림 5>의 complex 네트워크에 대한 하부 시스템의 자료는 다음과 같다.

제약식은 선형 제약식인 $\sum_{i=1}^{5} c_i x_i \le 20$이고, 시스템의 신뢰도의 함수식은 다음과 같이 주어 진다.

$$\begin{aligned} R_s(x) = {} & R_1(x_1)R_2(x_2)Q_3(x_3)Q_5(x_5) \\ & + Q_1(x_1)R_3(x_3)R_4(x_4)Q_5(x_5) \\ & + [R_1(x_1)R_3(x_3) + R_3(x_3)R_5(x_5) \\ & + R_5(x_5)R_1(x_1) - 2R_1(x_1)R_3(x_3)R_5(x_5)] \\ & \times [R_2(x_2) + R_4(x_4) - R_2(x_2)R_4(x_4)] \end{aligned}$$

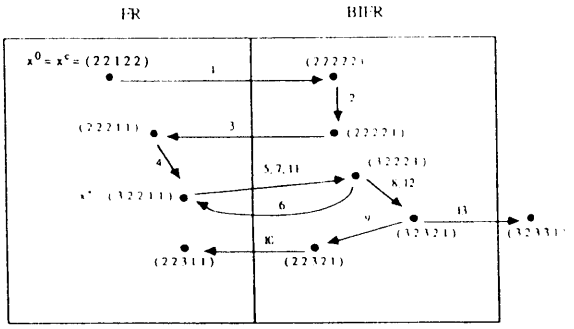
i	1	2	3	4	5
ri	0.70	0.85	0.75	0.80	0.90
$c1i$	2	3	3	3	1



<그림 5> **Bridge** 네트워크 시스템

*BIFR*의 경계 Δ_1은 $\Delta_1=c_{j(1)}\times 2=6$으로 주고, 초기해는 앞에서 언급한 바와 같이 two-phase 방법에 의해서 구했다. *EA*의 탐색과 정을 설명하면 다음과 같다. 이 탐색 과정을 알기 쉽게 <그림 6>에 타나내었다.

0. $\Delta_1=6$
1. $x^0=(2, 2, 1, 2, 2)$. $x=x^0$.
2. $x^c=x$. $E=F=\phi$.
3. $x=x(+3)=(2, 2, 2. 2, 2)$.
   $|b_1^c|=2<\Delta_1=6$.
4. $E=E\cup x=\{(2, 2, 2, 2, 2)\}$.
   $x\notin F$. $x=x(-5)=(2, 2, 2, 2, 1)$.
5. $x\notin FR$. 단계 4로 간다.
4. $E=E\cup x=\{(2, 2, 2, 2, 2). (2, 2, 2, 2, 1)\}$.
   $x\notin F$. $x=x(-4)=(2, 2, 2, 1, 1)$.
5. $x\in FR$. $x^f=x=(2, 2, 2, 1, 1)$.
   개선된 x^f는 $x^f=x(+1)=(3, 2, 2, 1, 1)$.
6. $R_s(x^f)=0.9932>R_s(x^c)=0.9765$.
   $x=x^f=(3, 2, 2, 1, 1)$.



<그림 6> 예제 1의 탐색경로(**EA**)

단계 2로 간다.

2. $x^c=x$. $E=F=\phi$.
3. $x=x(+4)=(2, 2, 2, 2, 1)$.
   $|b_1^c|=3<\Delta_1=6$.
4. $E=E\cup x=\{(3, 2, 2, 2, 1)\}$.
   $x\notin F$. $x=x(-4)=(3, 2, 2, 1, 1)$.
5. $x\in FR$. $x^f=(3, 2, 2, 1, 1)$. x^f는 개선되지 않음.
6. $R_s(x^f)=R_s(x^c)$,
   $x=x^c=(2, 2, 2, 1, 1)$.
   $F=F\cup E=\{(3, 2, 2, 2, 1)\}$.
   $E=\phi$. 단계 3으로 간다.
3. $x=x(+4)=(3, 2, 2, 2, 1)$.
   $|b_1^c|=5<\Delta_1=6$.
4. $E=E\cup x=\{(3, 2, 3, 2, 1)\}$.
   $x\in F$. $x=x(-1)=(3, 2, 2, 2, 1)$.
   $E=\phi$. 단계 3으로 간다.
3. $x=x(+3)=\{(3, 2, 3, 2, 1)\}$.
   $|b_1^c|=5<\Delta_1=6$.
4. $E=E\cup x=\{(3, 2, 3, 2, 1)\}$.
   $x\notin F$. $x=x(-1)=\{(2, 2, 3, 2, 1)\}$.
5. $x\notin R$. 단계 4로 간다.
4. $E=E\cup x=\{(3, 2, 3, 2, 1)\}, (2, 2, 3, 2, 1)\}$.
   $x\notin F$. $x=x(-4)=(2, 2, 3, 1, 1)$.
5. $x\in FR$. $x^f=(2, 2, 3, 1, 1)$. x^f는 개선되지 않음.
6. $R_s(x^f)=0.9923<R_s(x^c)=0.9932$,
   $x=x^c=(3, 2, 2, 1, 1)$.
   $F=F\cup E=\{(3, 2, 2, 2, 1)$,
   $(3, 2, 3, 2, 1), (2, 2, 3, 2, 1)\}$.
   $E=\phi$. 단계 3으로 간다.
3. $x=x(+4)=\{(3, 2, 2, 2, 1)\}$.
   $|b_1^c|=3<\Delta_1=6$.
4. $E=E\in x=\{(3, 2, 2, 2, 1)\}$.
   x$\in F$. $F=F\cup F=\{(3, 2, 3, 2, 1), (3, 2, 3, 2, 1)$,
   $(2, 2, 3, 2, 1)\}$.
   $E=\phi$. 단계 3으로 간다.
3. $x=x(+3)=(3, 2, 3, 2, 1)$.
   $|b_1^c|=5<\Delta_1=6$.
4. $E=E\cup x=\{(3, 2, 3, 2, 1)\}$.
   $x\in F$. $F=F\cup E=\{(3, 2, 2, 2, 1), (3, 2, 3, 2, 1)$,
   $(2, 2, 3, 2, 1)\}$.
   $E=\phi$. 단계 3로 간다.
3. $x=x(+4)=(3, 2, 3, 3, 1)$.
   $|b_1^c|=8>\Delta_1=6$. 단계 7로 간다.
7. 중단. $x^*=x^c=(3, 2, 2, 1, 1)$.

최적해는 $x^*=(3, 2, 2, 1, 1)$이며, $R_s(x^*)=0.9932$이다. 이 해는 Kohda/Inoue[9]에서 global 최적해임이 밝혀져 있다. Shi[18]등도 같은 최적해를 구하였다. 그런데, Aggarwal[1]은 global 최적해 보다 못한 $R_s=0.9921$인(2, 1, 3, 2, 1)만을 구하였다. 예제1에 대해 two - phase 방법에 의해 구한 각각 다른 초기해를 가지고 EA을 10번 시도한 결과, 6번의 lobal 최적해를 찾을 수 있었다(<표 1> 참조).

3.2 예제 2

다음 예를 Shi[18]에서 고려된 것으로서, complex 네트워크의 구조는 <그림 7>과 같다.

위의 구조에 대한 중복설계 문제는

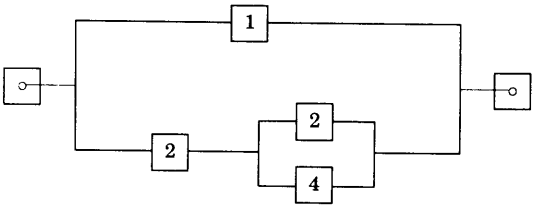
Maximize

$$R_s(x)=R_1(x_1)+Q_1(x_1)R_2(x_2)R_4(x_4)$$
$$+Q_1(x_1)R_2(x_2)R_3(x_3)Q_4(x_4)$$

subject to

<표 1> 예제 1의 10개의 초기해에 대한 최적해

초기해	$R_s(x^0)$	최적해	$R_s(x^*)$
1. (2 2 1 2 2)	0.9765	(3 2 2 1 1)	0.9932
2. (1 1 4 1 4)	0.9689	(2 2 3 1 1)	0.9923
3. (3 1 3 1 2)	0.9695	(3 2 2 1 1)	0.9932
4. (1 1 3 2 3)	0.9893	(2 2 3 1 1)	0.9923
5. (3 1 2 2 1)	0.9932	(3 2 2 1 1)	0.9932
6. (3 2 2 1 1)	0.9932	(3 2 2 1 1)	0.9932
7. (1 2 2 2 2)	0.9802	(2 2 3 1 1)	0.9923
8. (2 2 3 1 1)	0.9923	(2 2 3 1 1)	0.9923
9. (3 1 1 1 6)	0.9634	(3 2 2 1 1)	0.9932
10. (3 1 2 1 4)	0.9684	(3 2 2 1 1)	0.9932



<그림 7> Complex 시스템의 예

$\sum_{i=1}^{4} c1_i x_i \leq 30$

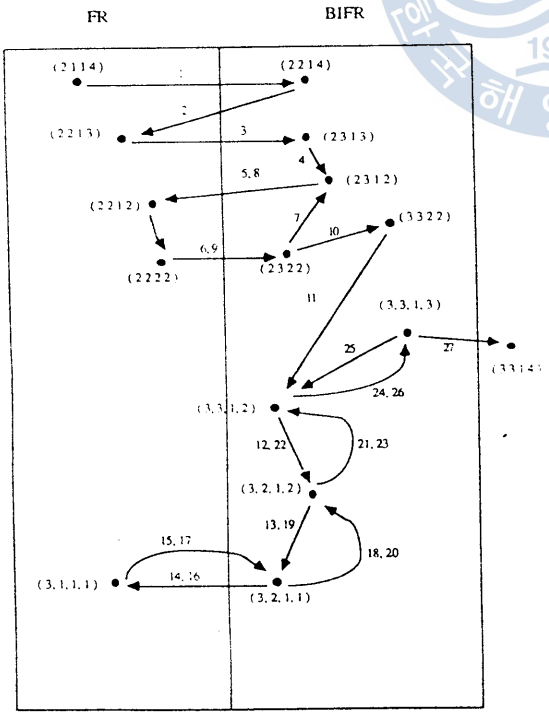
$\sum_{i=1}^{4} c2_i x_i \leq 40$

$x_i \geq 1$, 정수, $i=1, 2, 3, 4$.

로 주어진다. 그리고, 하부 시스템의 자료는 다음과 같다.

i	1	2	3	4
ri	0.80	0.75	0.70	0.65
$c1i$	6	4	3	2
$c2i$	9	4	4	3

*EA*을 위 문제에 적용하였으며, 이 때 *BIFR*의 경계 Δ_1과 Δ_2는 각각 $\Delta_1=2\times c1_{(1)}=12$, $\Delta_2=2\times c2_{(1)}=18$로 정했다. *EA*에 의해 구해진 최적해는 $x^*=(3, 1, 1, 1)$이였으며 $R_s(x^*)$는 0.9974이다. 탐색과정은 <그림 8>에 상세하게 나타나 있다. Complete enumeration에 의해 위의 해가 global 최적해임을 알 수 있었다. 이 문제에 대한 Shi[18]의 해는 $x=(2, 2, 1, 3)$, $R_s(x)=0.9970$이었다. 이 문제에 대해서 각각 다른 초기해로 *EA*을 10번 시도한 결과, 5번의 global최적해를 찾을 수 있었다(<표 2> 참조).



<그림 8> 예제 2의 탐색경로

<표 2> 예제 2의 10개의 초기해에 대한 최적해

초기해	$R_s(x^0)$	최적해	$R_s(x^*)$
1. (2 2 1 3)	0.9970	(3 1 1 1)	0.9974
2. (1 1 4 3)	0.9499	(1 3 2 3)	0.9961
3. (2 2 2 2)	0.9971	(3 1 1 1)	0.9974
4. (3 1 1 1)	0.9974	(3 1 1 1)	0.9974
5. (1 3 2 3)	0.9961	(1 3 2 3)	0.9961
6. (1 3 1 4)	0.9960	(1 3 1 4)	0.9960
7. (2 1 1 4)	0.9899	(3 1 1 1)	0.9974
8. (1 4 2 1)	0.9929	(1 3 2 3)	0.9961
9. (2 1 2 3)	0.9899	(3 1 1 1)	0.9974
10. (1 2 4 2)	0.9873	(1 3 2 3)	0.9961

4. 컴퓨터 실험결과

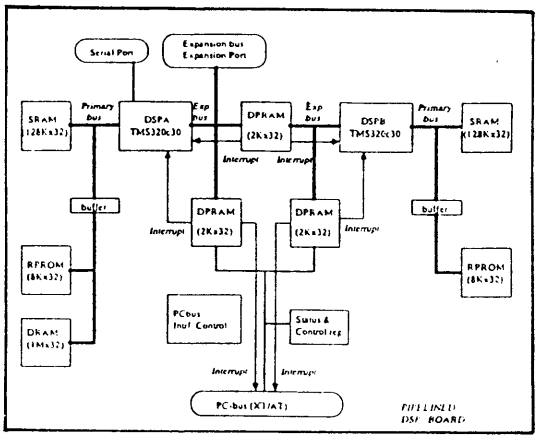
4. 1 고성능 다중프로세서에 의한 전용 시스템의 설계

본 연구에서는 신뢰도 중복설계 문제에서 발생되는 계산의 복잡성 및 반복성등의 문제점을 고성능 신호처리 프로세서로 구성되는 병렬처리 시스템을 이용하여 해결하고자 한다. 구현할 시스템은 TI(Texas Instrument)사의 DSP(Digital Signal Processor)인 TMS320C30 2개로 구성되는 알고리즘 처리 전용 Accelerator가 IBM - PC(486)에 연결되는 다중프로세서 구조로서 각 프로세서는 처리과정을 분담하게 된다. TI사의 TMS320C30은 33MFLOPs(Mega Floating Point Operation Per Sec)의 처리속도를 갖으며, 배열 연산에 있어 많은 시간을 차지하는 루프제어를 하드웨어적으로 수행하는 기능을 갖고 있어 알고리즘의 수행속도를 고속화 할 수 있다.

<그림 9>는 알고리즘 처리용 전용 Accelerator의 블럭도로서 두 개의 DSP는 Expansion Bus 및 Dual Port Memory를 이용하여 32비트의 통신경로를 구성하고 있으며, 각 DSP와 PC간의 통신경로는 8비트의 DPRAM을 이용하고 있다. 본 연구에서는 이러한 전용 시스템을 이용하여 시스템의 안전을 위한 신뢰도 중복설계 문제의 해결에 있어 처리의 고속화를 기하고자 한다.

4. 2 *EA*의 성능 비교

EA의 성능을 시험하기 위하여, <그림 10>의



<그림 9> **Algorithm** 처리 전용 **Accelerator**의 블럭도

n=7, 10, 15인 네트워크 구조를 고려했으며, <표 3>에 나타나 있는 12가지의 조합에 대하여 각각 10문제씩을 랜덤하게 구성하였다.

여기서, w_j는 b_j값을 정하기 위한 난수(random number)로서 $b_j = w_j \times \sum_{i=1}^{n} c_{ji}$이다.

각 문제는 모형(*PI*)와 같으며, 제약식은 다음과 같은 경우를 고려했다.

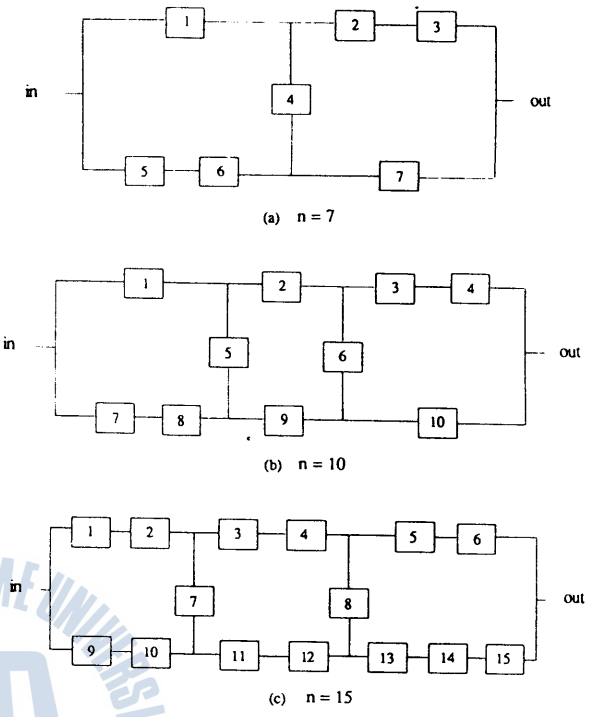
$$\sum_{i=1}^{n} c_{ji} x_i \le b_j, j = 1, 2, \cdots, m.$$

그리고, 신뢰도 함수식에 나타나는 하부시스템의 신뢰도 r_i는(0.6, 0.8), 제약식의 계수 c_{ji}는(0, 100)의 범위의 난수(uniform random number)로 각각 정하였다. 또 b_j는 $b_j = w_j \times \sum_{i=1}^{n} c_{ji}$로 계산되는데, w_j는(1.5, 2.5)의 다소 작은 범위와(2.5, 3.5)의 다소 큰 범위의 난수로 정하였다.

목적함수인 신뢰도 함수식 R_s는 recursive disjoint product 방법[12]에 의하여 프로그램하여 시스템의 네트워크 구조에 관한 자료만 입력되면 구할 수 있도록 하였다. 예를 들어, <그림 10>의 (a)에 대한 신뢰도 함수식은 부록 A에 나타나 있다.

본 연구에서 개발한 *EA*의 성능을 기존의 다른 발견적 방법(Shi, Kohda/Inoue)과도 비교하였다. 특히, Kohda/Inoue의 알고리즘은 국부 최적해에 빠지는 위험을 줄이고자 하는 EA와 같은 맥락에서 출발한 것이기 때문에 관심있는 비교 대상이며, Shi의 알고리즘은 위의 방법들과는 다르지만 컴퓨터 계산시간을 줄이기 위해 최단경로(minimal path)를 사용한 방법이므로 비교 대상에 포함시켰다. 또한 최근에 combinatiorial optimization 문제의 발견적 방법으로 비교적 자주 사용되고 있는 SA(Simulated annealing)방법도 비교 대상에 포함시켰다. 시스템의 중복설계 문제에 대한 SA의 상세한 단계는 부록 B에 수록되어 있다. 위의 4가지 발견적 방법(Shi, Kohda/Inoue, *EA*, *SA*)을 앞의 12개의 조합에 대하여 각각 10문제식(총 120문제)을 본 연구에서 설계한 다중프로세서에 의한 전용 시스템으로 시험하였다.

비교대상인 Kohda/Inoue[9]방법에서 Nakagawa/Nakashima[16]방법을 사용하여 초기해를 구했기 때문에, 동일한 조건에서 위의 4가지 방법



<그림 10> 전산실험을 위한 시스템의 구조

<표 3> 12가지의 조합

		n=7	n=10	n=15
m=1	1.5<wj<2.5	①	②	③
	2.5<wj<3.5	④	⑤	⑥
m=5	1.5<wj<2.5	⑦	⑧	⑨
	2.5<wj<3.5	⑩	⑪	⑫

<표 4> 발견적 방법의 성능 비교

	Problem Size n×m											
	7×1		7×5		10×1		10×5		15×1		15×5	
	W_j		W_j		W_j		W_j		W_j		W_j	
	1.5-2.5	2.5-3.5	1.5-2.5	2.5-3.5	1.5-2.5	2.5-3.5	1.5-2.5	2.5-3.5	1.5-2.5	2.5-3.5	1.5-2.5	2.5-3.5
Shi	A 0.01229 M 0.03718 O 3/10 T 0.02	A 0.00194 M 0.00542 O 0/10 T 0.02	A 0.02269 M 0.12476 O 2/10 T 0.03	A 0.00369 M 0.01495 O 0/10 T 0.04	A 0.05789 M 0.17357 O 0/10 T 0.11	A 0.00728 M 0.02621 O 0.10 T 0.18	A 0.01722 M 0.04065 O 1/10 T 0.08	A 0.00404 M 0.01092 O 0/10 T 0.13	A 0.09896 M 0.20495 O 0/10 T 0.37	A 0.02070 M 0.05475 O 0/10 T 0.59	A 0.07150 M 0.15827 O 1/10 T 0.14	A 0.01227 M 0.02365 O 0/10 T 0.23
Kohda/ Inoue	A 0.00312 M 0.02686 O 4/10 T 0.07	A 0.00007 M 0.00053 O 7/10 T 0.13	A 0.00317 M 0.01977 O 7/10 T 0.06	A 0.00033 M 0.00207 O 7/10 T 0.15	A 0.00027 M 0.00212 O 6/10 T 0.68	A 0.00008 M 0.00069 O 6/10 T 1.20	A 0.00282 M 0.01527 O 7/10 T 0.37	A 0.00015 M 0.00085 O 8/10 T 0.83	A 0.00086 M 0.00857 O 9/10 T 2.70	A 0.00001 M 0.00001 O 9/10 T 4.67	A 0.00148 M 0.01316 O 7/10 T 1.21	A 0.00016 M 0.00091 O 8/10 T 2.72
EA	A 0.00026 M 0.00138 O 6/10 T 0.12	A 0.00001 M 0.00010 O 9/10 T 0.18	A 0.00087 M 0.00871 O 9/10 T 0.10	A 0.00024 M 0.00217 O 8/10 T 0.16	A 0.0 M 0.0 O 10/10 T 0.91	A 0.0 M 0.0 O 10/10 T 1.29	A 0.00039 M 0.00386 O 9/10 T 1.52	A 0.0 M 0.00003 O 9/10 T 1.75	A 0.0 M 0.0 O 10/10 T 2.19	A 0.0 M 0.0 O 10/10 T 3.61	A 0.00085 M 0.00855 O 9/10 T 2.59	A 0.0 M 0.0 O 10/10 T 2.74
Simulated Annealing	A 0.00337 M 0.02786 O 6/10 T 0.85	A 0.00192 M 0.00515 O 2/10 T 0.78	A 0.00285 M 0.01977 O 8/10 T 1.18	A 0.00102 M 0.00390 O 4/10 T 0.84	A 0.01324 M 0.03261 O 2/10 T 3.26	A 0.02546 M 0.04472 O 0/10 T 3.83	A 0.00255 M 0.01527 O 6/10 T 3.56	A 0.00786 M 0.01682 O 0/10 T 3.66	A 0.02447 M 0.05579 O 3/10 T 6.98	A 0.05056 M 0.06631 O 0/10 T 9.03	A 0.00102 M 0.00855 O 8/10 T 6.79	A 0.01424 M 0.02301 O 0/10 T 8.99

들을 비교하기 위하여 각 알고리즘의 초기해는 Nakangawa/Nakashimap[16] 방법에 의해 결정하였다. 그러나, Shi의 방법은 "I - neighborhood" 내에서의 탐색방법이므로, Shi 방법만은 초기해를 (1, 1, ⋯, 1)로 구성하여 비교 하였다.

또한, *EA*의 *BIFR*은 모든 j에 대하여 $\Delta j = 2 \times c_j(1)$으로 설정하였고, Kohda/Inoue[9]방법에는 민감도 계수로서 0.5를 사용했다.

총 120문제에 대하여 위의 4가지 방법을 적용한 결과는 <표 4>에 수록 되어 있다. <표 4>는 다음과 같은 평가기준들에 의해 정리한 것이다.

$$A_i = \frac{1}{10}\sum_{j=1}^{10}(R_j^{\cdot} - R_{ij})/R_j^{\cdot}$$

$$M_i = \max_j \{(R_j^{\cdot} - R_{ij})/R_j^{\cdot}\}$$

O_i = 10문제 중에서 방법 i가 가장 좋은 해를 찾은 횟수

T = 평균 CPU time(sec)

여기서,

R_{ji} : 방법 i에 의한 j번째 문제의 시스템 신뢰도. $j = 1, 2, \cdots, 10$.

$R_j^{\cdot}$: j번째 문제의 global 최적해이거나, 4가지의 방법에 의해 찾은 R_{ij}의 최대치. $j = 1, 2, \cdots, 10$

<표 4>으로 부터 발견된 사항들을 정리하면 다음과 같다.

1. A. M. O는 생성된 해의 질을 나타내는 척도로 볼 수 있는데, 각각에 대하여, *EA*는 다른 방법들 보다 일관성 있게 나은 결과를 제공해 주고 있으며, Kohda/Inoue의 방법은 *EA*보다 다소 떨어지는 것을 알 수 있다. 그리고, Shi의 방법과 *SA*는 *EA*나 Kohda/Inoue방법에 비해 생성해의 질이 훨씬 떨어지는 것을 볼 수 있다.
2. 시간 T에 대해서는, Shi의 방법이 가장 적은 컴퓨터 수행시간이 소요되며, *SA*의 컴퓨터 수행시간이 제일 길다. 또, *EA*는 Kohda/Inoue보다 다소 많은 컴퓨터의 수행시간이 소요되는 것을 알 수 있다.

5. 결 론

본 연구에서는 비선형 정수계획의 새로운 발견적 해법인 EA(Excursion Algorithm)를 개발하였으며, 또한 EA를 시스템의 안전도를 높이기 위한 신뢰도 중복설계문제에 적용한 프로그램을 개발하였다.

EA의 성능을 비교하기위해 신뢰도 함수식을 disjoint product[12]방법에 의해 프로그램화하여 네트워크의 최단경로집합(minimal path set)만 입력하면 쉽게구할 수 있게 하였고, 적정한 규모의 중복설계문제를 고속으로 초리하기 위한 다중프로세서에 의한 전용시스템을 설계하여 기존의 발견적 해법들과 최근 인공지능분야에서 유용한 대안으로 등장하고 있는 SA알고리즘과 그 성능을 비교하였다. 비교 결과, EA가 기존의 방법들보다도 비교적 더 좋은 해를 생성하는 것을 관측하였다.

따라서, 신뢰도 중복설계문제나 이와 유사한 비선형 정수계획의 문제에 대한 알고리즘으로서 본 연구에서 개발한 EA는 유용한 대안이 된다고 사료된다.

6. 참고문헌

1) Aggarwal, K. K., "Redundancy optimization in general systems", IEEE Trans. Reliability, Vol R - 25, 330 - 332, 1976(Corrections, Vol. R - 26, 345, 1977).
2) Cerny, V., "A thermodynamical approach to the traveling salesman problem : An efficient simulation algorithm", J. Optimization Theory and applications, Vol. 45, 41 - 45, 1985.
3) Christopler, J. P., Wilhelm, W. W., "A perturbation scheme to improve Hiller's solution to the facilities layout problem", Management Science. Vol. 30, 1238 - 1249, 1984.
4) Cooper, M. W., "A survey of methods for pure nonlinear integer programming", Management Science, Vol. 27, 353 - 361, 1981.
5) Glover, F., "Future paths for I.P. and Links to A. I.", comput. & Ops. Res., Vol. 13, 553 - 549, 1986.
6) Glover, F., "Heuristics for integer programming using surrogate constraints", DecisionScience. Vol. 8, 156 - 166, 1977.
7) Hillier, F. S., "Quantitative toos for plant layout analysis", J. Indust. Enginneering, Vol. 14, 20 - 40, 1963.
8) Kirpatrick, S., Gelatt, C. d., Vecchi, M. P., "Optimization by simulated annealing", Science, Vol. 220, 671 - 680, 1983.
9) Kohda, T., Inoue, K., "A reliability optimization method for complex systems with the criterion of local optimality", IEEE Trans. Reliability, Vol. R - 31, 109 - 111, 1982.
10) Lawler, E.L. and Bell, M.D., "A method for solving discrete optimization problems", Operations Res., Vol. 14, 1098 - 1112, 1966.
11) Liggett, R. S., "The quadratic assignment problem : An experimental evaluation of solution strategies", Management Science, Vol. 27, 442 - 458, 1981.
12) Locks, M.O., "Recursive disjoint products : A review of three algorithms", IEEE Trans. Reliability, Vol. R - 31, 33 - 35, 1982.
13) Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., "Equation of state calculations by fast computiong machine", Chem. J. Phys., Vol. 21, 1087 - 1092, 1953.
14) Mitchell, T.J., "An algorithm for the construction of D - optimal experimental designs", Technometrics, Vol. 16, 203 - 201, 1974.
15) Morin, T.L. and Marsten, R.E., "An algorithm for nonlinear knapsack problems", Management Sciece, Vol. 22, 1147 - 1158, 1976.
16) Nakagawa, Y., Nakashima, K., "A heuristic method for determining optimal reliability allocation", IEEE Trans. Reliability, Vol. R - 26, 156 - 161, 1971.
17) Nakagawa, Y., Nakashima, K. and Hattori, Y., "Optimal reliability allocation by Branch and Bound techniques", IEEE Trans. Reliability, Vol. 27, 31 - 38, 1978.
18) Shi, D.H., "a new heuristic algorithm for constrainted redundancy optimization in complex systems", IEEE Trans. Reliability, Vol. R - 36, 621 - 623, 1987.

부록 A. Complex 시스템의 신뢰도 함수식

본 연구에서 개발한 EA에서는 complex 시스템의 신뢰도 함수식을 disjoint product 방법[12]에 의하여 구하였다. 그리고, 이를 프로그램화하여 네트워크의 최단 경로 집합(minimal path set)에 관한 자료만 입력하면 신뢰도 함수식을 구할 수 있도록 하였다.

<그림 10>의 (a)에 대한 신뢰도 함수식은 다음과 같이 구해진다.

$$
\begin{aligned}
R_s(\mathrm{x}) = {} & R_1(x_1)R_2(x_2)R_3(x_3) \\
& + R_1(x_1)Q_2(x_2)R_4(x_4)R_7(x_7) \\
& + R_1(x_1)R_2(x_2)Q_3(x_3)R_4(x_4)R_7(x_7) \\
& + Q_1(x_1)R_5(x_5)R_6(x_6)R_7(x_7) \\
& + R_1(x_1)Q_2(x_2)Q_4(x_4)R_5(x_5)R_6(x_6)R_7(x_7) \\
& + R_1(x_1)Q_2(x_2)Q_3(x_3)Q_4(x_4)R_5(x_5)R_6(x_6)R_7(x_7) \\
& + Q_1(x_1)R_2(x_2)R_3(x_3)R_4(x_4)R_5(x_5)R_6(x_6)R_7(x_7)
\end{aligned}
$$

부록 B. Simulated/annealing(SA) Algorithm

SA는 Metropolis[13]에 의해서 처음으로 창안되었으며, 이 알고리즘은 후에 VLSI의 설계문제나 Travelling Salesman Problem에 대한 해법으로 사용되었다. SA의 일반적인 단계는 Cerny[2]에 언급되어 있는데, 이를 근거로 본 연구에서 시스템의 중복 설계에 적용한 SA의 구체적인 단계는 다음과 같다.

0. 초기해 x^0를 구성한다. 초기온도 $T=0.001$.
   $x^c=x^0$, $x=x^c$.
1. itertion의 수 $N=1$.
2. $i=1$.
3. $j\neq i$인 [1, n]사이의 정수 j를 랜덤하게 선택한다.
   $x=x(+i)$. $x=x(-j)$. $D=R_s(x)-R_s(x^c)$.
4. $D>0$이면, 단계 5로 간다.
   $D\leq 0$이면 (0, 1) 사이의 난수(random unmber) U를 구한다.
   $U<e^{D/T}$이면, 단계 5로 간다.
   $U\geq e^{D/T}$이면, $i=i+1$. 단계 6으로 간다.
5. $x^c=x$, $i=i+1$.
6. $i>n$이면, 단계 7로 간다.
   $i\leq n$이면, 단계 3으로 간다.
7. $N=N+1$. $N>1000$이면, 단계 8로 간다.
   $N\leq 1000$이면, 단계 2로 간다.
8. $T=T/10$. $T<0.00005$이면, 단계 9로 간다.
   $T\geq 0.00005$이면, 단계 1로 간다.
9. 중단, $x^*=x^c$, $R_s(x^*)=R_s(x^c)$