海洋環境·安全學會誌 第8卷 第1號, pp.127~137 2002. 6. 30

論 文

신뢰도 최적화 문제에 대한 웹기반의 Solver 개발

김 재 환*

A Web-based Solver for solving the Reliability Optimization Problems

*J. H. Kim**

〈目 次〉

Abstract

This paper deals with developing a Web-based Solver, NRO(Network Reliability Optimizer) for solving three classes of reliability redundancy optimization problems which are generated in series systems, parallel systems and complex systems. Inputs of NRO consisted in four parts, that is, user authentication, system selection, input data and confirmation. After processing of inputs through internet, NRO provides conveniently the optimal solutions for the given problems on the Web-site.

To alleviate the risks of being trapped in a local optimum, HH(Hybrid-Heuristic) algorithm is incorporated in NRO for solving the given three classes of problems, and moderately combined GA(Genetic Algorithm) with the modified SA(Simulated Annealing) algorithm.

* 정회원, 한국해양대학교

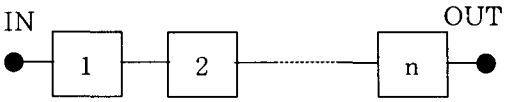
I. 서 론

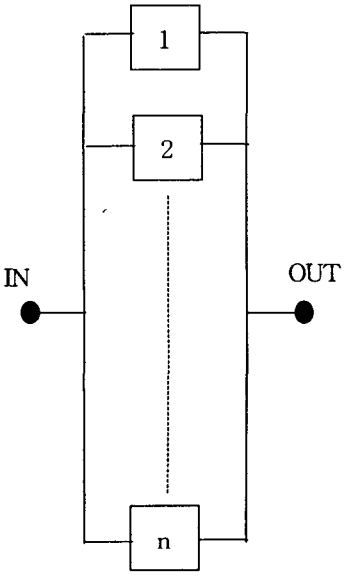
1.1 연구목적

인터넷은 미국 국방성에서 일정지역에 대한 폭탄 폭격과 같은 긴급 사태시에도 장애를 받지 않고 제 기능을 발휘할 수 있는 통신망 구축을 연구하여 네트워크를 개발했는데 이것이 바로 미국 국방성 최초의 연구 목적 네트워크인 ARPANET (1969)이다. 이 이후 연구 목적의 미 과학재단 네트워크인 NSFNET이 연결(1986)되었고 그 이후에는 일반 상업적인 목적의 네트워크가 연결(1990년이후)되면서 현재의 인터넷으로 발전을 이루었다. 인터넷의 초기에는 E-mail, Ftp, Newsgroup 등으로 학자들이나 전문인들이 사용하는 등 소규모로 사용되었으나 1991년 CERN의 WWW(World Wide Web) 서비스가 개발되어 일반인들도 쉽게 멀티미디어 정보를 제공할 수 있고 사용할 수 있게 되자 인터넷은 폭발적인 성장을 이루게 되었다. 지금은 전세계 수많은 호스트 컴퓨터가 인터넷에 연결되어 있는 거대한 네트워크가 되었고, 국내에서도 어디서나 손쉽게 접할 수 있을 정도로 인터넷의 보급이 급속하게 확대되었다.

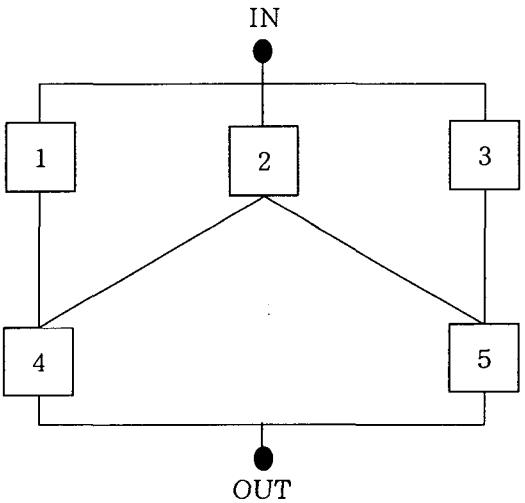
인터넷이 글로벌 정보 인프라의 기본 골격이 됨으로써, 인터넷은 이제 단순히 정보를 제공하고 공유하는 기능을 넘어서 인트라넷 어플리케이션이나 웹 기반의 정보 시스템 등과 같은 수단으로서 널리 이용되고 있다[10].

네트워크의 하드웨어 기술 및 클라이언트/서버 구조로서의 분산 컴퓨팅의 발달로 인해 인트라넷이나 인터넷 환경에서 다수의 컴퓨팅 자원을 접근하고 공유를 통해 연계 운영하게 하는 컴퓨팅 기술이 발달하게 되었다. 이에 소켓 프로그래밍, HTTP/CGI, 서블릿, RMI 뿐만 아니라 최근에는 CORBA, DCOM 등이 상업적으로 성공하여 표준이 됨으로써 분산 컴퓨팅 및 분산 객체 컴퓨팅 환경에서의 응용 프로그램 개발이 용이해졌다. 따라서, 인트라넷이나 인터넷 환경에서 클라이언트/서버로 구조화하여 의사결정을 지원하는 시스템에 대해 수리모형의 최적화 기법을 적용하거나, 경험적 접근방법, 시뮬레이션, 또는 통계적 분석 등의 수리 분석을 도입하는 것이 가능하게 되었다.





〈그림 1〉 직렬, 병렬 시스템



〈그림 2〉 콤플렉스 시스템

본 연구에서는 〈그림 1〉, 〈그림 2〉의 직렬, 병렬 시스템, 콤플렉스 시스템의 안전을 위한 하부시스템의 중복설계 최적해를 웹상에서 편리하게 제공하기 위해 현재까지 개발이 안된 웹기반의 solver인 NRO(Network Reliability Optimizer)를 개발하고자 한다. NRO의 자료입력 과정은 사용자 인증, 시스템 선택, 자료입력, 입력한 자료의 확인 등의 4단계로 구성된다. 인터넷을 통해 자료가 입력된 후에는 본 연구에서 적용한 효율적인 발견적 합성 해법(Hybrid-Heuristic Optimizer)에 의해 신뢰도 최적해가 웹상에서 제공된다.

1.2 연구문제 및 연구배경

본 연구에서 다루고자 하는 시스템의 신뢰도 중복설계는 〈그림 1〉과 〈그림 2〉와 같은 주어진 시스템 구조에 대해 자원이나 기술적 요인에 따른 다양한 제약하에서 하부시스템(subsystem)의 부품의 중복(redundancy)설계를 통하여 시스템의 신뢰도를 최적화하는 문제이다. 이러한 중복설계 문제는 다음과 같은 비선형 정수계획 모형 (P)로 나타낼 수 있다[15].

$$(P) \quad \text{maximize} \quad R_s(\boldsymbol{x})$$

$$\text{subject to}$$

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

$$x_i \ge 1, \text{ 정수}, \quad i=1, 2, \ldots, n$$

여기서, $R_s(\boldsymbol{x})$는 시스템의 신뢰도 함수, $g_{ji}(x_i)$는 i번째 하부시스템에 소요되는 j번째 자원의 양, b_j는 j번째 자원의 최대 사용가능량, x_i는 i번째 하부시스템의 부품의 개수를 각각 나타낸다.

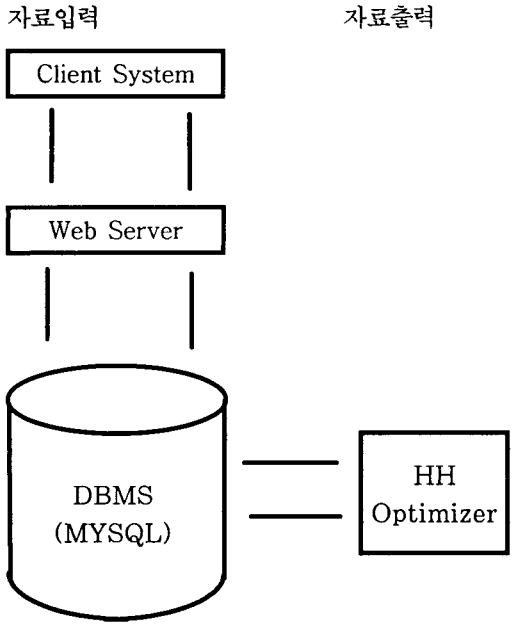
중복설계 문제에 대한 기존의 연구를 살펴보면, Sharma[12]와 Nakagawa[11]등은 직렬, 병렬 구조의 시스템에 대한 중복설계 문제를 다루었고, 콤플렉스 시스템에 대해서는 Aggarwal[2, 5], Gopal[7], Kohda[9], Shi[13], Kim[15, 16] 등에 의해 연구되었다. 중복설계 문제에 대한 정확한 해법(exact solution method)은 컴퓨터 수행시간이 지나치게 많이 소요되기 때문에 (NP-Complete), 많은 발견적 방법들이 대안으로 개발되었다([3], [4], [9], [13], [14], [16]). 이러한 방법들 중에서 Coit[14]의 GA(Genetic Algorithm)가 전역 최적해(global optimum)에 가장 가까운 해를 제공해 주는 것으로 알려져 있다[16]. 그러나, 이 문제에 대한 웹기반의 solver는 현재까지 개발이 안된 실정이다[17]. 따라서, 본 연구에서는 중복설계 문제에 대한 최적해를 웹상에서 편리하게 제공해 주는 solver인 NRO를 개발하고자 하며, NRO 성능의 향상을 위해 1차적으로 가장 좋은 최적해를 제공해 주는 것으로 알려져 있는 Coit[14]의 GA를 수행한 후 SA (Simulated Annealing) 알고리듬([1], [6], [8])으로 재 최적화시키는 발견적 합성해법을 NRO에 적용하였다.

II. 신뢰도 최적화를 위한 웹기반의 Solver

본 장에서는 모형 (P)에 대해 본 연구에서 개발한 웹기반의 solver인 NRO의 개요와 지역 최적해(local optimun)에 도달하는 위험을 줄이기 위해 적용된 발견적 합성 해법에 대해 언급하고자 한다.

2.1 NRO의 개요

NRO는 스크립트 언어인 PHP를 이용하여 웹상에서 사용자가 필요로 하는 자료를 입력하면 Web Server를 이용하여 DBMS인 MySQL에 값을 저장한다. 입력자료가 저장되면 NRO는 C언어로 구현된 HH Optimizer를 MySQL과 연동시키기



〈그림 3〉 NRO의 개요

위해 C API(Application Programing Interface)를 이용하여 출력결과인 최적해를 웹상에 제공한다. NRO의 자료 입력과정은 사용자 인증, 시스템의 선택, 자료입력, 입력한 내용의 확인 등의 4단계로 구성되며 상세한 과정은 3장에서 다루고자 한다.

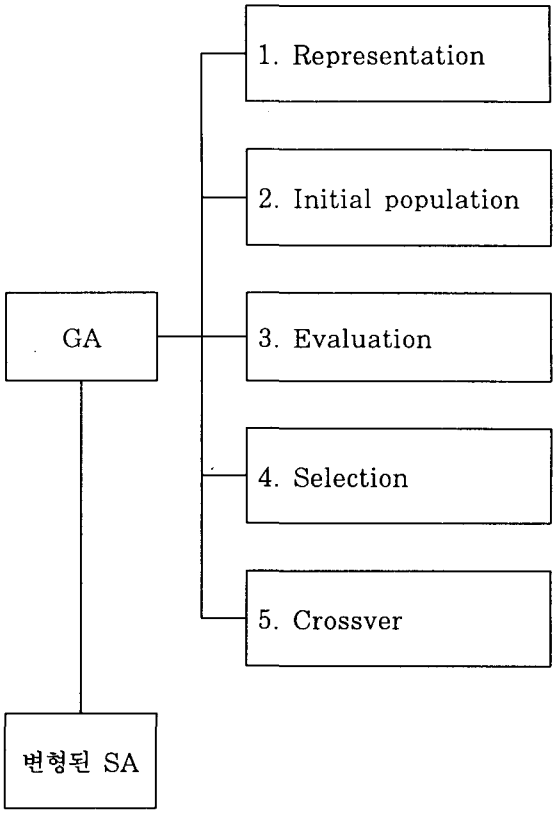
2.2 발견적 합성해법(Hybrid-Heuristic Optimizer)

NRO에서는 신뢰도의 최적해를 구하기 위해 1차적으로 가장 좋은 최적해를 제공해 주는 것으로 알려져 있는 Coit[14]의 GA를 수행한 후 본 연구에서 고안한 변형된 SA 알고리듬으로 재 최적화 시키는 발견적 합성해법을 적용하였다.

이 알고리듬은 〈그림 4〉와 같이 GA와 변형된 SA 알고리듬으로 구성된다.

1장의 모형(P)의 문제에 대한 발견적 합성해법의 구체적인 절차는 다음과 같다.

step 1. pop_size, GTOT, STOT를 결정한다. num = 1.



〈그림 4〉 발견적 합성 해법의 구성

step 2. 초기해의 집단을 pop_size만큼 랜덤하게 생성한다.

step 3.(Evaluation) 생성해 $\boldsymbol{x}_c$ ($c = 1, 2, \ldots\ldots,$ pop_size)에 대한 적합도(fitness)를 다음과 같이 계산한다.

$$eval(\boldsymbol{x}_c) = \begin{cases} R_S(\boldsymbol{x}_c): \boldsymbol{x}_c\text{가 문제}(P)\text{의 가능해 일때.} \\ R_S(\boldsymbol{x}_c) - M: \text{문제}(P)\text{의 가능해가 아닐때.} \end{cases}$$

(여기서, M은 $0 < M < 1$인 penalty 계수이다.)

step 4.(Selection)

4.1 개체군에 대해서 적합도의 합 및 평균을 계산한다.

$$F = \sum_{k=1}^{pop\_size} eval(\boldsymbol{x}_c), \quad c = 1, 2, \ldots\ldots, \text{pop\_size}$$

$$\overline{F} = \frac{F}{pop\_size}$$

4.2 각각의 생성해 x_c에 대해서 기대값 E_c를 계산한다.

$E_i = \frac{eval(x_c)}{\overline{F}}$, $c = 1, 2, \ldots\ldots,$ pop_size

4.3 각각의 생성해 x_c에 대해서 실제값 A_c를 구한다.

$A_c = \lfloor E_c + 0.5 \rfloor$, $c = 1, 2, \ldots\ldots,$ pop_size

4.4 실제값 A_c의 값만큼 다음 세대의 생성해 x_c를 구한다.

step 5.(Crossover) 서로 다른 두 개의 생성해 x_c를 선택한 후 임의의 교배점을 선택하여 교배점의 오른쪽 유전자를 교환한다.

step 6. num $> GTOT$ 이거나 모든 $eval(x_c)$값이 같으면, step 7로 간다.
그렇지 않으면, step 3으로 간다.

step 7.(변형된 SA)

7.1 num1 = 1.

7.2 $x_c' = x_c$.

$0 \le i \ne j \le n$인 i, j를 랜덤하게 선택한다.

7.3 r을 0 또는 1중에서 랜덤하게 선택한다.

7.4 $r = 0$이면,

$x_c' = (x_1', x_2', \ldots, x_i' - 1, \ldots, x_j' - 1, \ldots, x_n')$

그렇지 않으면,

$x_c' = (x_1', x_2', \ldots, x_i' + 1, \ldots, x_j' + 1, \ldots, x_n')$

7.5 $eval(x_c') > eval(x_c)$ 이면 $x_c = x_c'$.

7.6 num1 =num1+1. num1 > STOT이면, step 8로 간다. 그렇지 않으면, step 7.2로 간다.

step 8. 최적해 $x^* = x_c'$.

위의 발견적 합성해법은 성능 향상을 위해 기존의 GA를 수행한 후, 본 연구 에서 고안한 변형된 SA알고리듬으로 재 최적화시키기 때문에, 기존의 GA 보다 항상 더 좋은 최적해를 제공해 준다. 이러한 결과에 대한 예제는 3장에 나타나 있다.

III. NRO의 구현 예제

본 장에서는 NRO를 구현하기 위한 서버사양 및 자료 입력과정과 발견적 합성해법의 구체적인 절차를 예제를 통해 설명하고자 한다.

3.1 서버 사양

NRO을 구현하기 위한 대한 서버 및 소프트웨어에 대한 사양은 다음과 같다.

서버	: Pentium(r) pro processor, 128.0 MB RAM
OS	: Redhat LINUX 6.0 kernel 2.2.9-6kr
Web 서버	: Apache 1.3.9
Backend	: PHP 3.0.12
DBMS	: MySQL 3.22.22

3.2 자료 입력 과정

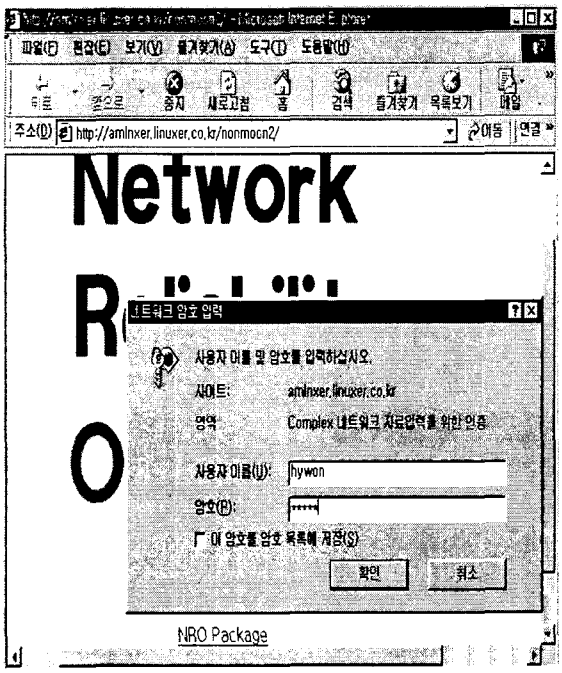
각 사용자가 본 연구에서 개발한 NRO를 이용하기 위해서는 인터넷을 이용하여 홈페이지에 접속한 후 다음과 같은 4단계의 자료입력 과정을 거쳐야 한다.

i) 사용자 인증
ii) 시스템 선택
iii) 자료입력
iv) 입력한 내용의 확인

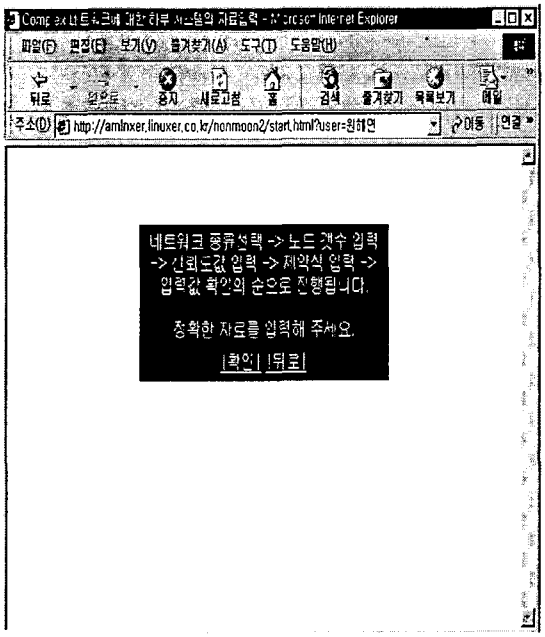
i) 사용자 인증

첫 번째 과정인 사용자 인증(User Authentication)은 DBMS인 MySQL을 이용하여 구성하였다. 서버측 스크립트에 Header( )함수를 사용하면 웹서

버에서는 〈그림 5〉의 메시지를 클라이언트 브라우저에게 보낸다.



〈그림 5〉 사용자 인증

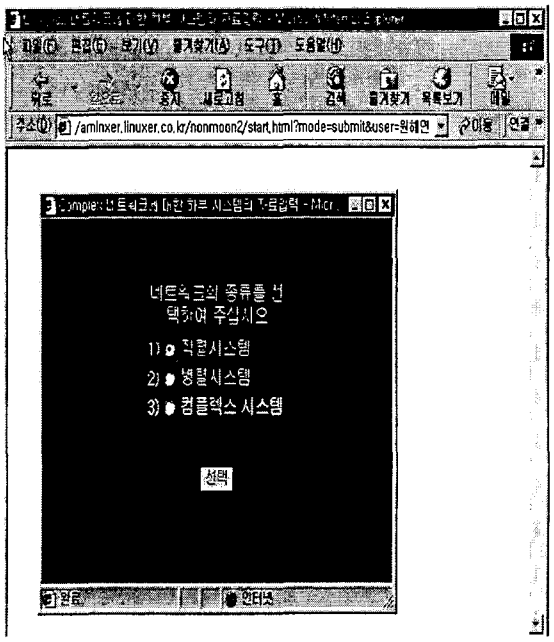


〈그림 6〉 자료입력 과정

여기에 사용자 아이디, 암호를 입력하게 되면 각각 $PHP_AUTH_USER, $PHP_AUTH_PW 변수에 들어가게 되고 이 값을 가지고 MySQL에 SQL문을 이용하여 인증과정을 처리한다. 사용자가 입력한 아이디와 암호가 확인되면 〈그림 6〉의 자료입력 과정의 화면이 나타나게 된다.

ii) 시스템의 선택

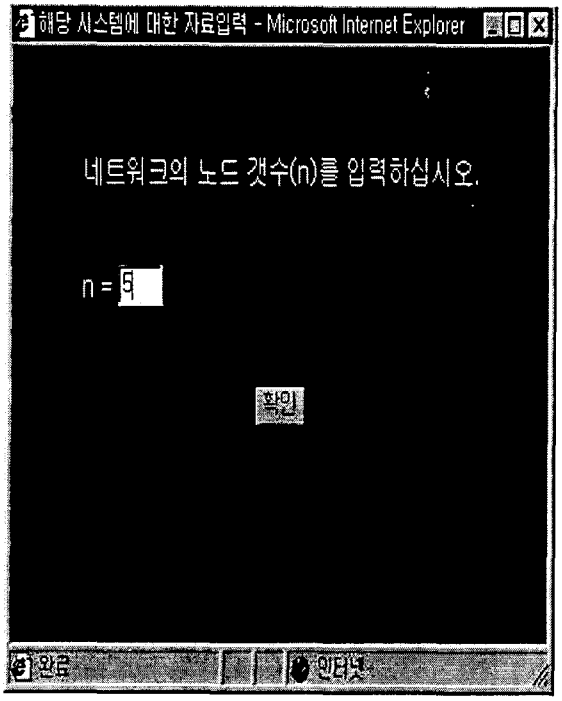
두 번째 해당 시스템의 선택과정은 사용자 인증을 거친 후 나타나는 〈그림 6〉의 화면에서 확인 버튼을 누르면 〈그림 7〉의 화면이 나타나고 입력하려는 해당 시스템을 선택하면 된다.



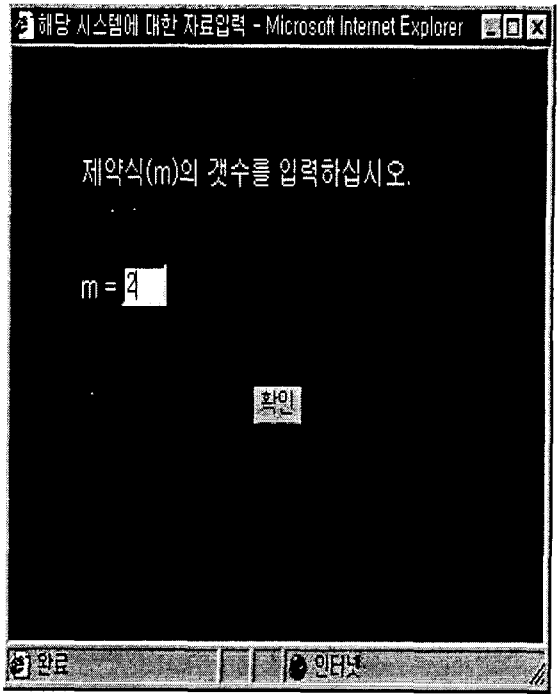
〈그림 7〉 시스템 선택

iii) 자료입력

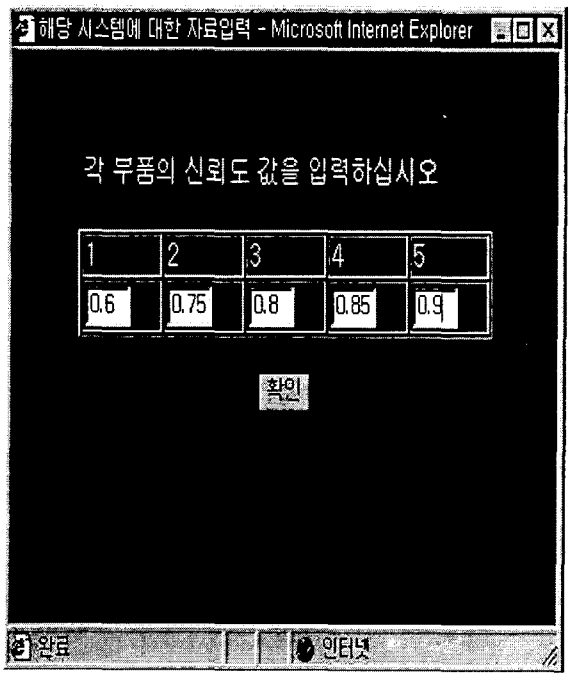
세 번째 자료입력 과정은 직렬시스템과 병렬시스템의 경우 노드의 갯수(n) 입력(〈그림 8〉) ⇒ 신뢰도 값 입력(〈그림 9〉) ⇒ 제약식(m)갯수 입력(〈그림 10〉) ⇒ 제약식 입력(〈그림 11〉)의 4단계 과정으로 이루어지며, 컴플렉스 시스템의 경우에는 노드의 갯수입력 다음에 incidence matrix 입력(〈그림 12〉) 과정이 첨가되었다.



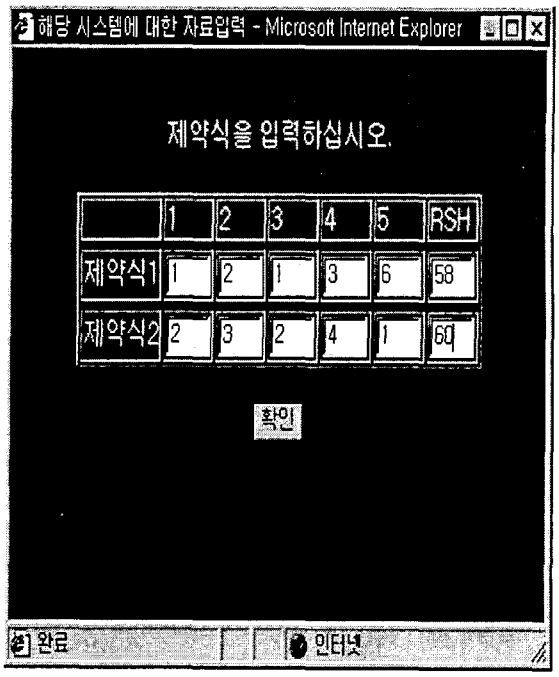
〈그림 8〉 노드의 갯입력



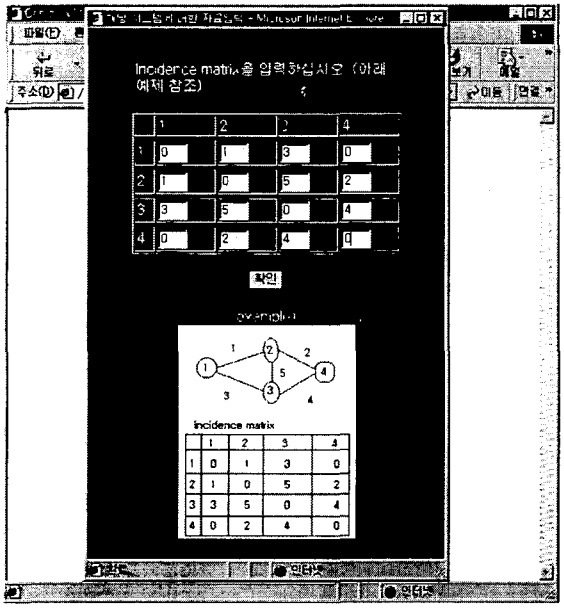
〈그림 10〉 제약식 갯수 입력



〈그림 9〉 신뢰도 값 입력



〈그림 11〉 제약식 입력



〈그림 12〉 incidence matrix 입력

iv) 입력한 내용의 확인

마지막으로 세 번째 자료입력과정까지를 마치면 지금까지 입력한 값을 확인(〈그림 13〉)함으로서 4가지의 자료입력 과정을 모두 마치게 된다.

〈그림 13〉 입력 값 확인

3.3 예 제

NRO의 입력과정인 〈그림 7〉의 시스템 선택에서 〈그림 12〉의 컴플렉스 시스템을 선택하고 〈그림 8〉에서 〈그림 11〉까지의 과정에서 〈표 1〉과 같은 자료를 입력했을 때의 신뢰도 최적화 문제는 다음과 같다[15].

〈표 1〉 하부시스템의 부품의 자료 :

$n = 5,\ m = 1,\ \mathrm{RHS}=20$

i	1	2	3	4	5
신뢰도	0.70	0.85	0.75	0.80	0.90
제약식	2	3	2	3	1

최대화 $R_S(\boldsymbol{x}) =$
$R_1(x_1)R_2(x_2)Q_3(x_3)Q_5(x_5)$
$+Q_1(x_1)R_3(x_3)R_4(x_4)Q_5(x_5)$
$+[R_1(x_1)R_3(x_3)+(R_3(x_3)R_5(x_5)$
$+R_5(x_5)R_1(x_1)$
$-2R_1(x_1)R_3(x_3)R_5(x_5)]$
$\times[R_2(x_2)+R_4(x_4)-R_2(x_2)R_4(x_4)]$

제약식 : $\sum_{i=1}^{5} c_i x_i \le 20$

$x_i \ge 1,\ i=1,\ 2,\ 3,\ 4,\ 5$

여기서 $R_i(x_i) = (1-(1-r_i)^{x_i})$로 주어지고 $Q_i(x_i) = 1-R_i(x_i)$이며, r_i는 부품 i의 신뢰도 값이다. 위의 제약식으로부터 $x_1 \le 6,\ x_2 \le 4,\ x_3 \le 6,\ x_4 \le 4,\ x_5 \le 10$ 이므로 각각의 x_i를 이진수로 표현한 문자열 길이의 총합은 16비트이다. pop_size = 10이라 하면 랜덤하게 생성된 초기해의 집단은 다음과 같다.

$\boldsymbol{x}_1$ = [1010010010010001]
$\boldsymbol{x}_2$ = [0010010110010001]
$\boldsymbol{x}_3$ = [1111111110111101]
$\boldsymbol{x}_4$ = [0111011111111001]
$\boldsymbol{x}_5$ = [1111110110110011]
$\boldsymbol{x}_6$ = [1110110111011011]
$\boldsymbol{x}_7$ = [0011111111111001]
$\boldsymbol{x}_8$ = [0010010010010011]
$\boldsymbol{x}_9$ = [0011111111110001]
$\boldsymbol{x}_{10}$ = [0111011011110101]

생성해 $\boldsymbol{x}_c(c=1,\ldots,10)$의 적합도를 계산할 때 제약식을 만족하지 않는 생성해 $\boldsymbol{x}_c$에 대해서는 다음과 같은 penalty 계수를 부여한다.

$$eval(x_c) = \begin{cases} R_S(x_c) : x_c\text{가 가능해 일때.} \\ R_S(x_c) - 0.9 : x_c\text{가 가능해가 아닐때.} \end{cases}$$

위에서 계산된 적합도(fitness)를 가지고 각 생성해 x_c가 다음 세대에 나타날 기대값을 계산하면 〈표 2〉와 같다.

〈표 2〉 초기해의 기대값 계산

initial population	x value	$R_S(x)$	$\frac{f_i}{\sum f}$	$\frac{f_i}{f}$	actual count
1. [1010010010010001]	(3,1,1,1,1)	0.966387	0.27	2.74	3
2. [0010010110010001]	(1,1,3,1,1)	0.960302	0.27	2.74	3
3. [1111111110111101]	(7,7,7,3,13)	0.100000	0.03	0.28	1
4. [0111011111111001]	(3,5,7,7,9)	0.099998	0.03	0.28	0
5. [1111110110110011]	(7,7,3,3,3)	0.099997	0.03	0.28	0
6. [1110110111011011]	(7,3,3,5,11)	0.099995	0.03	0.28	0
7. [0111111111111001]	(1,7,7,7,9)	0.099982	0.03	0.28	0
8. [0010010010010011]	(1,1,1,1,3)	0.897191	0.25	2.55	3
9. [0011111111110001]	(1,7,7,7,1)	0.099981	0.03	0.28	0
10.[0111011011110101]	(3,5,5,7,5)	0.099974	0.03	0.28	0
Sum		3.523870	1.0	10.0	
Average		0.352381	0.1	1.0	
Max		0.966387	0.27	2.74	

〈표 2〉에서 계산된 actual count값에 따라 새롭게 생성된 개체는 〈표 3〉과 같다.

〈표 3〉 새로운 개체 생성

	Mate	crossover site		New population	$R_S(x)$
1.	[1010010010010001]	4	8	[1010010010010001]	0.966387
2.	[1010010010010001]	7	7	[1010010110111101]	0.098762
3.	[1010010010010001]	9	2	[1010010010010011]	0.969380
4.	[0010010110010001]	1	8	[0010010110010001]	0.960302
5.	[0010010110010001]	8	1	[0010010010010011]	0.897191
6.	[0010010110010001]	10	10	[0010010110010011]	0.965402
7.	[1111111110111101]	2	7	[1111111010010001]	0.099995
8.	[0010010010010011]	5	1	[0010010110010001]	0.960302
9.	[0010010010010011]	3	2	[0010010010010001]	0.891312
10.	[0010010010010011]	6	10	[0010010010010001]	0.891312
Sum					7.700346
Avg					0.770035
Max					0.969380

위와 같은 GA의 과정을 100번 수행한 후, 변형된 SA과정을 1000번 수행하여 얻어지는 신뢰도 최적해는 〈표 4〉와 같다. 이 예제에 대한 최적해는 $\boldsymbol{x}^* = (3, 2, 2, 1, 1)$, $R_S(\boldsymbol{x}^*) = 0.9932$이며, 2장에서 언급한 바와 같이 발견적 합성해법이 기존의 GA에 의해서 구해진 해 보다 모든 경우에 대해 더 좋은 최적해를 찾아주는 것을 알 수 있다.

〈표 4〉 신뢰도 최적해

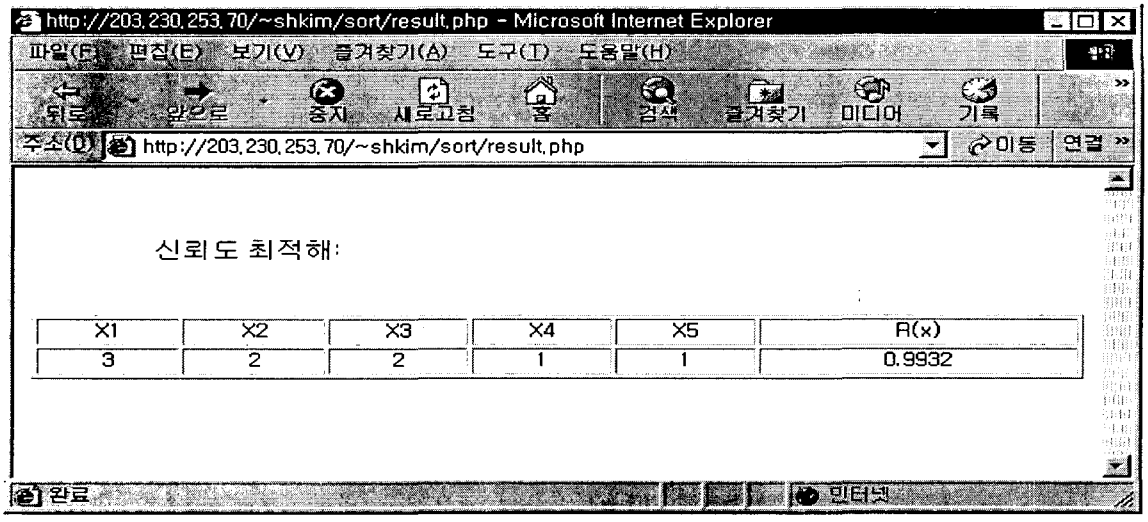
	GA로 구한 해(x_c)	HH의 최적해(x^*)
1	0.9654	0.9932
2	0.9634	0.9932
3	0.8973	0.9689
4	0.9198	0.9932
5	0.9690	0.9932
6	0.9634	0.9932
7	0.9211	0.9932
8	0.9690	0.9932
9	0.9932	0.9932
10	0.9243	0.9932

발견적 합성해법에 의해 최적해가 구해지면 NRO는 〈그림 14〉와 같은 화면으로 최적해를 웹상에서 제공한다. 〈그림 14〉에 제시된 신뢰도 최적해는 부품1, 2, 3, 4, 5의 개수를 각각 3개, 2개, 2개, 1개, 1개로 중복 설계했을 때, 시스템의 신뢰도가 0.9932가 된다는 것을 의미한다. 부품 1, 2, 3, 4, 5의 개수를 각각 3개, 2개, 2개, 1개, 1개로 중복 설계했을 때, 시스템의 신뢰도가 0.9932가 된다는 것을 의미한다.

IV. 결 론

본 연구에서는 직렬 시스템, 병렬 시스템, 콤플렉스 시스템에서 발생하는 신뢰도 중복설계 문제에 대한 웹기반의 solver인 NRO를 개발하였다. NRO의 자료 입력과정은 사용자 인증, 시스템 선택, 자료입력, 입력한 내용의 확인 등의 4단계를 거치며 모두 Web-site 상에서 처리된다. 모든 데이터의 입력 과정이 끝나면, NRO의 최적화 절차인 발견적 합성 해법에 의해 신뢰도 최적해가 웹상에서 편리하게 제공된다. NRO 성능의 향상을 위해 1차적으로 가장 좋은 최적해를 제공해 주는 것으로 알려져 있는 Coit[14]의 GA를 수행한 후 SA 알고리듬으로 재 최적화시키는 발견적 합성해법을 NRO에 적용하였다.

앞으로의 연구과제는 직렬-병렬 시스템등의 보다 다양한 시스템을 포함시켜 NRO를 범용성이 있는 solver로 개발하는 것이다.



〈그림 14〉 신뢰도 최적해

참 고 문 헌

[1] Cerny, V., "Thermodynamical Approach to the Traveling Salesman Problem" JOTA, vol 45, 41-51, 1985.

[2] Aggarwal, K.K., "Redundancy optimization in general systems", IEEE Trans. Reliability, vol R-25, 330-332, 1976.

[3] Glover, F., "Future paths for I.P. and Links to A.I.", Comput, & Ops. Res., vol 13, 553-549, 1986.

[4] Glover, F., "Heuristics for integer programming using surrogate constraints", Decision Science. vol 8, 156-166, 1977.

[5] Aggarwal, K.K., Gupta, J.S., Misra, K.B., "A new heuristic criterion for solving a redundancy optimization problem", IEEE Trans. Reliability, vol R-24, 86-87, 1975.

[6] Kirpatrick, S., Gelatt, C.D., & Vecchi, M.P., "Optimization by simulated annealing", Science, vol 220, 671-680, 1983.

[7] Gopal, K., Aggarwal, K.K., Gupta, J.S., "An improved algorithm for reliability optimization", IEEE Trans. Reliability, vol R-27, 325-328, 1978.

[8] Metropolis, N., "Equation of State Calculations by Fast Computing Machines", Journal of Chemical Physics, vol 21, 1087-1092, 1953.

[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] Intranets Redefine Corporate Information System, http: //home. -netscape.com/comprod/at_work/white_paper/indepth.html, 1996

[11] Nakagawa, Y., Nakashima, K., "A heuristic method for determining optimal reliability allocation", IEEE Trans. Reliability, vol R-26, 156-161, 1971.

[12] Sharma, J., Venkateswaran, K.V., "A direct method for maximizing the system reliability", IEEE Trans. Reliability, vol R-20, 255-259, 1971.

[13] Shi, D.H., "A new heuristic algorithm for constrainted redundancy optimization in complex systems", IEEE Trans. Reliability, vol R-36, 621-623, 1987.

[14] Coit, D.W., Smith, A.E., "Reliability optimization of series-parallel systems using a genetic algorithm", IEEE Trans. Reliability, vol 45, 254-266, 1996.

[15] Kim, J.H., Yum, B.J., "A heuristic method for solving redundancy optimization problems in complex systems", IEEE Trans. Reliability, vol 42, 572-578, 1993.

[16] Kim, J.H., "A study on a global optimization method for solving redundancy optimization problems in series-parallel systems", 해양환경.안전학회지, 제6권, 제1호, 23-33, 2000.

[17] Kuo, W., "Optimal reliability design Fundermentals and applications", Cambridge University Press, 2001.