

(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(51) Int. Cl.⁶
G06F 15/18

(11) 공개번호 특1999-0077006
(43) 공개일자 1999년 10월 25일

(21) 출원번호	10-1998-0705136	(87) 국제공개번호	WO 1997/32261
(22) 출원일자	1998년 07월 03일	(87) 국제공개일자	1997년 09월 04일
번역문제출일자	1998년 07월 03일		
(86) 국제출원번호	PCT/US1996/02758		
(86) 국제출원출원일자	1996년 03월 01일		
(81) 지정국	EP 유럽특허 : 오스트리아 벨기에 스위스 독일 덴마크 스페인 프랑스 영국 그리스 이탈리아 룩셈부르크 모나코 네덜란드 포르투갈 스웨덴 핀란드		
	국내특허 : 아일랜드 캐나다 일본 대한민국 멕시코 노르웨이		
(71) 출원인	워즐 윌리엄 피.		
	미국 미시간 48160 미란 웨스트 메인 스트리트 214		
(72) 발명자	워즐 윌리엄 피.		
	미국 미시간 48160 미란 웨스트 메인 스트리트 214		
(74) 대리인	박장원		

심사청구 : 없음

(54) 유전자 프로그래밍방법 및 시스템

요약

본 발명은 유전자 프로그래밍기술을 이용하여 프로그래밍문제를 해결하기 위해 컴퓨터에 구현되는 방법과 시스템에 관한 것이다. 적합함수는 첫째 해결법과 둘째 해결법의 비교우위를 측정한다. 유전자 프로그래밍시스템은 그래프 감소연산자를 포함하는 복수의 프로그램유전자스트링을 생성한다. 각각의 프로그램유전자스트링은 프로그래밍문제의 가능한 해결법을 나타낸다. 각각의 유전자스트링을 위한 해결법을 생성하기 위해 입력데이터가 각각의 프로그램 유전자스트링에 적용된다. 각 프로그램 유전자스트링은 생성된 해결법과 적합함수를 비교함으로써 평가된다. 프로그램 유전자스트링은 적합도에 따라서 진화되고, 유전자스트링은 종결기준이 충족될 때까지 되풀이하여 진화된다.

명세서

기술분야

본 발명은 유전자 프로그래밍기술을 통해 효율적인 프로그램을 만드는 컴퓨터시스템에 관한 것으로, 특히 본 발명은 유전자알고리즘과 그래프감소기술을 합성하여, 프로그램의 효율적인 전개를 가능케하고, 입력이 알려져있고 프로그램을 서로 비교, 판단하는 기능을 가지는 컴퓨터시스템을 생산하는 기술에 관한 것이다.

배경기술

오늘날 대부분의 컴퓨터들은 본 노이만(von Neumann)형으로 알려진 컴퓨터들이다. 이러한 컴퓨터들은 주로 어드레스나 라벨로 식별되는 메모리로부터, 데이터에 동작이 적용되는 CPU로의 데이터의 이동을 요구한다. 이러한 데이터는 자주 원래의 위치 또는 다른 메모리의 위치로 다시 이동한다. 이러한 데이터의 계속적인 움직임은 비효율적이고, 이러한 장치를 만드는 능력을 제한한다.

람다계산법은 원하는 결과를 얻기 위해 정보를 처리하는 다른 방식을 보여주고 있다. 람다계산방식에서, 특별한 '람다식' 데이터와 기능을 밀접하게 연결하는 구조를 설명하기 위해 만들어졌다. 이러한 식을 사용함으로써, 노이만형 컴퓨터에 의해 생성되는 결과와 같은 결과를 생성하는 일반적인 프로그램장치가 구성될 수 있다.

D. A. Turner에 의해 쓰여진 글 ('A New Implementation Technique for Applicative Languages', Software - Practice and Experience, vol. 9, no. 1, pp. 31-49, 1979)에서는 람다식을 컴비네이터(Combinator)라 불리는 더욱 간단하고 강한형으로 추출하여 컴퓨터구조를 구성하는 방법을 설명하고 있다. 컴비네이터는 1930년대에 Schonfinkel과 Curry에 의해 처음으로 소개되었고, Turner는 이 컴비네이터를 컴퓨터구조를 구성하는 한 방법으로서 처음으로 제안한 사람이다. 그의 뒤를 이어 Clarke, Gladstone, MacLean, Norman 등이 상기 구조의 구현을 쉽게 할수있도록 만들어지는 컴퓨터하드웨어를 자세히 설명하였다. 이 후, 이에 해당하는 컴비네이터 하드웨어를 설명한 미국 특허 4,734,848을 포함하여 여러 가지의 자료들이 컴비네이터에 관하여 쓰여졌다.

컴비네이터는 그래프감소시스템의 한 예이다. 즉, 프로그램이 그래프감소연산자의 한 셋트에 의해서 변형되는 요소들의 리스트로서 표현되는 시스템이다. 컴비네이터의 간결성과 입증된 보편성 때문에, 이 발명

은 컴비네이터에 중점을 두고 있으며, 그래프를 나타내는 스트링에 작용하는 구조를 포함, 모든 그래프감소구조의 사용을 다루고 있다.

컴비네이터 K, S, I, B, C, W 는 아래의 함수에 의해 정의되고 도 1의 그래프로 나타낸다. 각각의 경우 f, g, x, y 는 함수, 연산자, 상수 또는 컴비네이터식이다.

$$Kxy = x$$

$$Sfgx = fx(gx)$$

$$Ix = x$$

$$Bfgx = f(gx)$$

$$Cfxg = (fx)g$$

$$Wfx = (fx)x$$

만약 프로그램은 트리의 각 가지가 프로그램의 갈라진 부분을 나타내는 방식의 트리구조로서 나타낼 때, 컴비네이터는 구조를 바꾸기위해 프로그램트리를 직접적으로 변경함으로써, 트리를 평가하는 결과를 변화시킨다. 컴비네이터 연산시스템은 변수에 관련된 사항을 제거하기위해 주로 브래킷 추출을 사용하였다 - 즉 매 시간 프로그램을 변경하는 입력데이터가 실행된다. 브래킷 추출법은 입력 데이터에 관련되지않는 순수한 컴비네이터식을 생성한다. 대신, 컴비네이터는 직접적으로 그래프나 트리으로써 배열되는 입력 데이터에 적용된다.

John Holland 는 1975년 그의 논문 Adaptation in natural and artificial system 에서 연산의 문제를 해결하기위해 유전자를 사용하는 방법을 설명하였다. Holland 는 임의로 선택된 초기값의 여러 가지 세트로부터 하나의 값이 다른 값보다 우수하다는 것을 판단하는 방법이 있다는 것을 제공하였다. 해결책은 유전자를 조합함으로써 얻어질 수 있고 이러한 유전자의 값은 메카니즘을 통해 나타난다.

이러한 유전자 알고리즘은 유효한 값들의 큰 모집단에서 원하는 값들을 수렴하는 방법을 제공한다. 만약 많은 유효값중의 우수한 하나의 값이 구하여지거나, 값들이 2 진수 스트링과 같은 일반적 형태로써 규정되면, 이러한 스트링을 다른 값과 비교할 때 하나의 값의 질을 나타내는 값을 할당할 수 있다는 것을 보여주는 후보자의 비교적 작은 풀로 부터 선택, 연결, 경우에 따라서 변화하여 빠르고 효율적으로 최선의 해결점을 찾아 나아갈 수 있다.

유전자알고리즘은 주로 프로그램에서 설명되고, 이 프로그램은 알려진 측정함수로부터 최대 또는 최적의 값을 구한다. 예를 들어, 많은 유효경로중에서 여행의 가장 효율적인 경로를 찾는 '순회 세일즈맨 (Traveling Salesperson)' 문제는 유전자 알고리즘의 문제를 설명하기 위해 자주 사용된다.

유전자알고리즘기술을 사용하여 프로그램을 생성하기 위한 과거의 시도는 복잡한 구조를 만드는 방법을 제공하는데 유전자 알고리즘과 룰베이스시스템의 조합을 포함하였다. 이것은 2 진수 값을 룰에 매치하고, 문제해결을 위해 룰의 최적조합을 찾기위해 그 값을 안출하여 프로그램된다. 상기 내용은 Holland 와 Burks 1987 미국특허 4,697,242 에 기술되어있다.

또다른 과거의 시도에서, 유전자 알고리즘의 원리는 LISP 프로그램과 퍼텐셜 유전자 요소로서의 프로그램을 설명하고, 유전자알고리즘룰을 이용하여 이러한 유전자요소로부터 더욱 복잡한 LISP 프로그램을 인출한다. 상기의 기술내용은 J. R. Koza's 의 유전자프로그램에 관한 책과 여러 가지의 유전자프로그램에 관한 특허에 기술되어있다 (US patents 4,935,877. 5,136,686. 5,343,554). 상기의 특허내용은 유전자스트링이 다양한 길이를 갖고, 또한 유전자와 프로그램단편과의 일대일 매핑에 의한 컴퓨터식에 관련될수있다고 제안하였다. 따라서, Koza 에 의한 시스템은 새로운 프로그램을 생성하기 위해 유전자 알고리즘에 사용되는 유전자값과 LISP 식의 매핑을 요구한다.

발명이이루고자하는기술적과제

따라서, 본 발명의 목적은 기성하드웨어로 구성될수있고, 또한 소프트웨어 장치로 구현이 가능한, 종래의 문제를 해결할 수 있는 프로그램을 만들기 위해 유전자프로그램과 그래프감소기술을 사용하는 컴퓨터시스템을 제공하는데 있다. 이러한 시스템은 그래프감소 연산자스트링의 형태로 나타나는 더욱 효율적인 프로그램을 계속적으로 생성하기위해 유전자의 문제를 이용함으로써 얻을 수 있다.

본 발명은 프로그램표현의 기초적인 방법으로서 어떠한 그래프 감소시스템의 이용도 가능하다. 그래프 감소시스템은 프로그램이 데이터 또는 프로그램구성요소의 그래프를 변형 또는 감소하는 일련의 연산자에 의해 표현되는 시스템이다. 상기 시스템은 스트링과 스트링조작에 기초한 그래프감소 시스템으로 표현되는 그래프를 포함한다.

그래프의 간결함, 보편성, 단순성등으로 인해, 컴비네이터에 기초한 그래프 감소시스템은 단일 유전자스트링으로 나타나고, 본 발명에서는 매핑이 요구되지 않는다. 프로그램 유전자스트링을 입력데이터에 적용하고, 결과적인 출력의 품질 또는 적합성을 측정함으로써 각각의 프로그램의 스트링이 평가된다.

본 발명은 프로그램 유전자스트링 자신이 프로그램인 유전자 프로그램장치를 만드는 종래의 시도와는 다르다. 종래의 기술에서는 유전자스트링과 프로그램단편이 매핑되어졌다. 유전자스트링의 이러한 매핑은 시험을 위한 후보프로그램을 어셈블하기 위해서 다양한 메모리로의 계속적인 데이터전송을 요구하기 때문에 시간이 많이 소비된다. 컴비네이터의 사용을 통해서 본 발명은 유전자스트링의 매핑과 매핑과 관련된 반복적인 데이터전송의 필요성을 제거한다. 따라서, 본 발명에 따른 유전자 프로그램시스템은 데이터와 변수들의 매핑을 해야하는 종래의 유전자프로그램시스템 보다 빠르게 작동하는 특성을 지닌다.

도면의 간단한 설명

도 1은 다양한 컴비네이터의 의해 수행된 함수를 나타낸 그래프.

도 2는 본 발명에 따른 컴비네이터 스트링을 싣고, 평가하는 컴비네이터 장치의 블록도.

도 3은 병렬로 연결된 복수의 컴비네이터 장치를 포함하는 유전자 프로그램시스템의 블록도.

도 4는 본 발명에 따른 유전자프로그램 시스템에 의한 과정순서도.

도 5는 원하는 입력데이터를 입력하는 방법을 설명한 순서도.

도 6은 유전자 스트링의 초기풀을 생성하는 방법을 설명한 순서도.

도 7은 트리구조의 입력데이터 세트를 나타낸 그래프.

도 8은 유전자 프로그램시스템으로서의 작동이 가능한 일반적 컴퓨터의 블록도.

발명의구성및작용

도 2 에서 보는바와 같이, 메인버스(30)는 다양한 메모리와 다른시스템의 요소 사이의 데이터흐름의 경로를 제공한다. 제어버스(32)는 컴비네이터장치의 다양한 구성요소들을 연결하고, 요소들 사이에 제어명령을 제공한다. 컴비네이터 평가부/제어부(CEU/제어부)(34)는 상기의 버스(30)(32)과 전기적으로 연결되어 있다. 상기 CEU/제어부(34)는 컴비네이터 프로그램 유전자스트링을 평가한다. 마이크로코드메모리(36)는 각각의 컴비네이터를 평가하는 내용을 포함하는 ROM이고, 다양한 컴비네이터 프로그램유전자스트링을 평가하는데 필요한 룰과 명령을 제공한다.

또한, CEU/제어부(34)는 평가되는 유전자스트링의 다양한 요소를 꺼내올수있고, 상기의 버스(30)(32)와 전기적으로 연결되어있는 I/O 프로세서(IOP)(38)와 산술논리연산부(ALU)(40)를 제어한다. 제어버스(32)는 IOP(38) 과 ALU(40)의 동작을 제어하는 CEU/제어부(34)에 의해서 이용된다. IOP(38)은 컴비네이터장치와 다른 유전자프로그램시스템사이에서 인터페이스를 제공한다. 상기 ALU(40)은 2+3, 3-4 등의 간단한 수학적식과 테스트를 평가하는데 이용된다.

주메모리(42) 와 스택메모리(44)는 메인버스(30)와 연결되어있다. 주메모리(42)는 CEU/제어부(34)에 의해 평가되는 프로그램 유전자스트링과 프로그램이 적용되는 입력데이터를 포함한다. 스택메모리(44)는 프로그램유전자스트링을 평가한 결과의 중간상태와 값을 저장한다. 예를 들어, 스택메모리(44)는 중간평가가 완성되는동안 평가가 중단되었던 곳을 되부르는 포인터를 저장하기도 한다.

도 2 가 컴비네이터장치의 실시예를 보이고있지만, 이같은 구조물은 종래의 전통적인 컴퓨터시스템에서 하드웨어를 작동하는 소프트웨어 형태로 구현될 수 있다. 소프트웨어 임플리멘테이션을 사용할 때, 시스템의 CPU는 도 2의 ALU (40)의 기능을 수행한다. CEU/제어부(34)와 ALU(40)은 필요할 때 이용되는 소프트웨어 라이브러리의 개별적인 프로그램에 의해서 대체된다. 주메모리(42) 와 스택메모리(44)는 시스템의 RAM 안의 메모리블록으로서 구현되고, 시스템의 I/O 처리시스템은 컴비네이터식과 데이터의 입력과 출력을 위해 IOP(38)을 대신한다. 평가프로그램은 특정 유전자스트링의 요소를 계속적으로 구하고, 평가를 실행하는 적절한 컴비네이터 라이브러리 서브루틴을 부르는 마이크로코드(36)를 대신하여 이용된다.

컴비네이터와 유전자프로그램의 본질적으로 비슷한 특성때문에, 본 발명은 병렬 프로세스시스템에서 이용하기 적합하다. 도 3은 병렬로 연결된 복수의 컴비네이터장치를 활용한 유전자프로그램시스템의 블록도이다. 복수의 컴비네이터장치(50)는 병렬로 연결되어 있고, 각각의 컴비네이터장치(50)는 하드웨어 (도2에 도시) 또는 소프트웨어로 구현될 수 있다.

임의의 수로 이루어진 컴비네이터장치(50)는 병렬로 이루어지며, 컴비네이터장치의 수는 여러 가지의 다양한 요인들에 의해서 좌우된다. 먼저, 비용의 한계가 컴비네이터장치의 수를 제한할 수 있다. 사용되는 컴비네이터장치의 수가 많을수록 시스템의 비용은 더욱 늘어난다.

속도요구조건은 특정시스템에서 사용되는 컴비네이터장치의 수를 제한할 수 있다. 빠른 측정을 요구하는 시스템은 낮은 측정속도를 가지는 시스템보다 더 많은 수의 컴비네이터장치를 필요로 한다. 또한, 해결해야할 문제의 복잡성이 컴비네이터장치의 수를 결정할 수 있다. 비교적 단순한 문제가 복잡한 문제보다 적은 수의 컴비네이터장치를 필요로 할 것이다.

마지막으로 유전자프로그램 시스템에 이용되는 컴비네이터장치의 수는 유전자스트링의 수에 의존한다. 각 세대에서 많은 유전자스트링을 평가해야하는 시스템은 많은 수의 컴비네이터장치를 이용함으로써 이득을 얻을 것이다. 예를 들어, 각 세대마다 100개의 유전자스트링을 평가해야하는 유전자 프로그램시스템은 100개의 컴비네이터장치를 갖는다. 즉, 한 개의 컴비네이터장치가 한 개의 유전자스트링을 평가한다. 이러한 시스템에서는 100개의 유전자스트링이 모두 동시에 평가되고, 시스템이 더 빠르게 동작하도록 한다.

도 3 에서와 같이, 각각의 컴비네이터장치(50)는 데이터/제어버스(46)와 프로그램유전자스트링버스(48)에 연결되어 있다. 유전자 프로그램제어기(52)는 상기 데이터/제어버스(46)에 연결되어있다. 유전자 프로그램제어기(52)는 평가작업을 컴비네이터장치(50)에 분배함으로써 유전자프로그램시스템의 전반적인 동작을 제어하고, 컴비네이터장치들로부터 결과를 구하고, 다양한 유전자스트링을 진화시킨다.

도 8 은 본 발명과 함께 사용할 수 있는 일반적인 목적의 컴퓨터의 블록도로써, 버스(10)는 다양한 시스템구성요소들을 연결하고, 데이터, 명령어 등의 흐름을 위한 공통의 통로를 제공한다. 중앙처리부(CPU)(12)는 버스(10)와 연결되어있고, 실제연산작업을 수행한다. 랜덤엑세스메모리(14) 또한 버스(10)와 연결되어있고, 데이터와 다른 정보를 저장하기 위한 위치를 제공한다. 데이터저장장치(16)는 버스(10)와 연결되어있고, 비휘발성의 정보를 저장한다. 데이터 저장장치(16)는 디스크드라이브, 테이프드라이브 또는 이와 비슷한 저장장치의 사용이 가능하다.

입력장치(18)는 버스(10)와 연결되어있고, 컴퓨터사용자가 데이터, 명령어 또는 다른 정보를 컴퓨터시스템에 입력할수있도록 한다. 입력장치(18)는 키보드, 광스캐너, 마이크론 또는 읽기가능한 신호를 세대할 수 있는 다른 장치일 수 있다. 사용되는 입력장치(18)는 적용방법에 따라 다양하게 만들어질 수 있다. 만약 시스템이 손으로 쓴 글을 알아보는데 사용되었다면, 입력장치는 반드시 사람의 글씨를 읽거나 디지털화할 수 있어야한다. 이때, 입력장치로서 광스캐너, 감압성의 쓰기자리판, 광판독기를 갖는 광전펜 등

이 이용될 수 있다.

어떤 상황에 따라서, 복수의 입력장치가 필요한 경우가 있다. 예를 들어, 음성인식시스템에서 이용되는 유전자프로그램시스템은 일정한 음성패턴을 입력하기 위한 마이크로폰과 다양한 사용자정의 파라미터를 입력하기 위한 키보드와 같은 장치를 필요로 한다.

도 8 에서 보는바와 같이, 디스플레이장치(20)는 버스(10)를 통하여 다른 구성소자들과 연결되어있다. 상기 디스플레이장치(20)는 사용자가 정의한 파라미터, 프로그램동작의 상태, 동작결과 등을 디스플레이하는데 이용되는 비디오모니터이다. 버스(10)에 연결되어 있는 출력장치(22)는 프로그램결과를 출력한다.

유전자프로그램시스템의 의한 진행과정의 전체적동작은 도 4 에 도시되어있다. 단계 58 에서 유전자프로그램시스템이 유전자스트링을 진화시키기전에, 다양한 파라미터와 데이터가 반드시 시스템에 제공되어야 한다.

도 5 에서, 스텝 86 에서는 사용자가 유전자폴에 포함될 유전자스트링의 수를 입력한다. 다음 스텝 88 에서 사용자는 초기 유전자스트링의 원하는 길이나 길이의 범위를 입력한다. 스텝 90 에서는 사용자는 유전자스트링의 변화의 도수를 입력하고, 스텝 91 에서는 유전자스트링의 치환도수를 입력한다. 유전자스트링의 교배율에 관한 정보는 스텝 92 에서 입력된다.

위에서 설명한바와 같이, 데이터 입력단계 86-92 에서는 시스템의 사용자에게 의해서 데이터가 입력될 수 있다. 입력된 데이터는 도 3의 GP 메모리(54)와 같은 메모리장치에 저장된다. 상기와 다른 방법으로 86-92 단계에 입력되는 데이터는 영구적으로 메모리장치에 저장될 수 있고, 따라서 사용자에게 의한 입력을 필요로 하지않을 수도 있다. 사용자가 일정한 단계에서 데이터를 입력하지 않으면, 디폴트값이 사용된다. 마지막으로, 86-92 단계에서 입력되는 데이터는 주어진 파라미터값의 범위안에서 유전자프로그램시스템에 의해서 임의로 결정된다. 예를 들어, 변환확률은 .001 과 .01 사이의 값에서 임의로 선택될 수 있고, 초기 유전자스트링의 길이는 40 개릭터와 100 사이에서 선택될 수 있다.

도 5 에서 보는바와 같이, 스텝 94 에서 사용자는 각각의 프로그램 유전자스트링에 이용될 입력데이터를 입력한다. 상기에서 설명한바와 같이, 상기 입력데이터는 해결해야하는 문제에 따라 다양하게 변화한다. 입력데이터는 유전자 프로그램시스템의 사용을 위해 메모리에 저장된다.

스텝 95 에서는 사용자가 상수로 사용될 값을 입력한다. 이후에 설명할 유전자스트링 세대에서 상수는 단일문자값, 문자 또는 수의 스트링일 수 있다. 이 스텝95 에서의 상수입력의 리스트는 프로그램유전자스트링을 구성하는데 사용될 것이다.

스텝 96 에서 사용자는 메모리에 저장되어있는 적합함수를 입력한다. 적합함수는 하나의 결과가 다른 결과보다 더 나은지를 판단하는데 사용된다. 적합함수를 선택하고 적용하는 것에 관한 더 자세한 사항은 이후에 서술하겠다.

스텝 98 에서, 종결판단의 세트가 사용자에게 의해서 입력된다. 결과판단은 유전자스트링의 평가가 종결되어야 할 때를 판단하는데 사용된다. 종결판단을 결정하는것에 관한 자세한 사항은 아래에서 설명될 것이다.

도 4 를 참고로 하여, 스텝 59 에서는 초기유전자폴이 생성된다. 유전자폴은 컴비네이터를 이용하여 구성된 프로그램 유전자스트링으로 구성되어있다. 상기 스텝 59 에서는 컴비네이터, 연산자, 상수의 스트링의 임의의 생성을 필요로 한다. 도 6 에서, 초기 유전자폴을 생성하는 첫 번째 스텝 100 은 초기 유전자폴의 크기에 관한 파라미터를 구하는데 관계한다. 초기 유전자폴의 크기는 주로 프로그램의 복잡성의 평가와 후보자폴의 다양성의 필요성에 따라서 사용자에게 의해서 결정되나, 디폴트값 또는 임의로 생성된 값을 사용하여 결정될 수 있다. 스텝 102 에서는 유전자폴이 가득차있는지를 판단한다. 즉, 폴의 유전자스트링의 수가 스텝 100 에서 결정된 폴의 크기와 같은지를 판단한다. 초기에 유전자폴은 비어있고, 루틴은 다음 유전자스트링의 길이가 결정되는 스텝 104 까지 루틴이 실행된다. 유전자스트링의 길이는 임의로 결정되거나, 사용자에게의해서 선택될 수 있다. 바람직하게는 길이의 범위는 초기폴을 위한 다양한 범위의 후보자를 생성하는데 사용된다. 예를 들어, 첫 번째 스트링이 30개 구성요소의 길이이고, 다음은 43 개, 그 다음은 22 개의 길이이다.

스텝 106 은 유전자스트링의 길이가 충분한지를, 즉 유전자스트링의 길이가 스텝 104 에서 결정된 길이와 같은지를 판단한다. 초기에 유전자스트링의 생성은 시작하지않았고, 유전자가 임의로 선택되는 스텝 108 까지 루틴이 실행된다. 선택된 유전자는 컴비네이터, 상수, 연산자중의 하나이다. 유전자의 선택은 컴비네이터에 주어진 우선권과 가중된 임의선택에 의해 결정될 수 있다.

컴비네이터가 스텝 110 에서 선택되면, 컴비네이터는 스텝 118 에서 유전자스트링의 결과에 더하여진다. 컴비네이터가 스트링의 첫 번째 유전자이면, 유전자스트링을 구성하는 시작포인트를 형성한다.

스텝 112 에서 상수가 선택되면, 상수는 스텝 118 에서 유전자스트링의 결과에 더하여진다. 상기 컴비네이터와 같이, 상수가 스트링의 첫 번째 유전자이면, 유전자스트링을 구성하는 시작포인트를 형성한다.

프로그램 유전자스트링의 상수의 표시는 특정하게 적용되나, 일반적으로 연산자와 상수를 뚜렷이 구별하는 방법을 포함해야한다. 예를 들어, 문자 'C' 의 상수값이 유전자스트링에 포함된 상수의 일부분일 때, 이는 컴비네이터 'C' 와 뚜렷이 구분되어야한다.

문자의 상수값과 컴비네이터를 구별하는 방법은 여러 가지가 있다. 바람직한 실행은 문자상수가 작은 따옴표안에 있는 문자로 정의되고 (예를 들어 'C'), 컴비네이터가 부호없는 문자 C 로 정의된 (예를 들어 C) 프로그램언어의 규정을 사용한다. 이와 비슷하게, 상수를 형성하는 문자스트링은 큰 따옴표안에 있고 (예를 들어 'Fred'), 숫자는 임의의 소수점과 부호를 가진 0-9 사이의 부호없이 쓰인다 (예를 들어 -3.14159). 스트링은 문자상수의 서브트리로서 프로그램 유전자스트링안에 포함되어있으나 (예를 들어 ('F' 'r' 'e' 'd')), 큰 따옴표안에 상수리스트에 포함될 수 있다 (예를 들어 'Fred').

프로그램 유전자스트링에 명확하게 상수를 포함하는 다른 방법으로서, 사용자는 프로그램 유전자스트링이 적용되는 데이터의 일부분과 같이 유용한 값을 입력할 수 있다. 예를 들어, 수학적 값을 계산할 프로그램

은 pi, e 등의 수학적 상수들을 유용하게 사용한다. 화학적물질을 분석하는 프로그램은 화학상수를 포함한다. 프로그램유전자스트링이 적용되는 데이터의 분기로서 이러한 상수를 포함함으로써, 프로그램 유전자스트링이 이러한 상수를 선택하고 필요한 프로그램을 생성하는데에 선택한 상수를 사용하는 동작을 할 수있게 한다고 가정한다. 이러한 접근방법으로, 프로그램유전자스트링을 단순화되어서 컴비네이터와 연산자만을 포함하게된다. 그러나, 본 발명의 바람직한 실시예로는 도 6 에서 설명된 상수가 직접적으로 프로그램유전자스트링에 포함된 경우이다.

스텝 118 에서 유전자가 유전자스트링에 더하여진뒤에, 루틴은 스트링이 충분히 긴지를 판단하기위해 스텝 106 으로 진행된다. 원하는 길이를 얻을때까지 유전자는 반복적으로 유전자스트링에 더하여진다. 유전자스트링이 완성되면, 프로그램은 생성된 식의 괄호가 서로 균형을 이루는 스텝 107 로 이동한다.

상기 동작을 필요로 하는 까닭은 괄호의 쌍이 유전자스트링의 트리구조를 설명하는 연산자를 구성하기 때문이다. 예를 들어, 유전자스트링 ((2 3 + K)(4 7 *) 2 1) 은 2 와 1 을 갖는 두 개의 분기와 트리의 서로다른 분기에 각각의 스트링 2 3 + K 와 4 7 * 를 형성하는 두 개의 서브트리를 갖는다.

유전자스트링이 생성되면, 열림괄호 '(' 와 닫힘괄호 ')' 는 유전자스트링 입력리스트의 서로 다른 종류로서 구별되어, 스텝 106 의 끝에서 생성된 유전자는 반드시 검사되어, 만약 여분의 열림괄호가 있으면 스트링에 열림, 닫힘수의 균형을 맞추기 위해 여분의 열림괄호 수에 해당하는 닫힘괄호가 스트링의 끝에 추가되어진다. 마찬가지로, 닫힘괄호의 수가 더 많을 때는 열림괄호가 스트링의 앞쪽에 추가된다. 반면, 균형을 맞추는 다른 방법으로는 동시에 스트링에 임의의 장소에서 열림괄호와 닫힘괄호를 추가하는 방법이다. 상기 방법은 식에 필요한 구조를 추가하는 우수한 방법이다.

스텝 107 이후에, 본 발명에 따른 시스템은 유전자풀이 가득차있는지를 판단하기위해 스텝 102 로 되돌아간다. 만약 유전자풀이 가득차있는 경우에는 초기 유전자풀의 생성이 완성된 것이다. 그렇지 않은 경우는 상기에서 설명된 스텝을 따라서 다른 유전자스트링이 생성되어진다.

도 4 를 참고하여, 컴비네이터스트링의 초기 유전자풀을 생성한 뒤, 입력데이터에 각 수를 적용하고 (스텝 60), 적합함수를 상기 스텝 60 에서 생성된 결과에 적용하여 적합도를 판단함으로써 (스텝 62) 초기생성결과가 평가되어진다.

스텝 60 에서는 입력데이터의 적용은 결과를 생성하기위해 유전자풀의 각 유전자스트링에 입력데이터를 적용함으로써 이루어진다. 초기의 유전자스트링의 첫 번째 생성은 전적으로 임의로 생성된 유전자로만 이루어져있고, 이후의 유전자풀은 이전의 생성에서 진화된 유전자스트링으로 구성되어있다. 데이터에서 한 개이상의 컴비네이터를 감소하기위해 사용자에게 의해서 공급된 입력데이터는 각 스트링에 적용된다. 컴비네이터의 감소는 값을 생성한다.

간단한 예로서 만약 프로그램유전자스트링이 수를 제공하는 컴비네이터식이고 프로그램스트링이 입력숫자 5 에 이용되었다면, 그 결과는 25 이다.

이 스텝에서는 연산자가 다른 타입의 데이터에 적용될 때 일어날 상황을 고려하는 것이 필요하다. 예를 들어, A+3 과 같은 컴비네이터스트링이 유전자프로그램시스템에 의해서 어떻게 평가되는가의 판단이 반드시 실행되어야한다. 바람직하게는 '+' 연산자의 정의가 'D' ('A' 앞의 세 글자)의 결과를 생성하기위해 확장된다. 상기 정의를 사용하여 연산자의 기능은 시스템에서 정의된 전체데이터에 걸쳐서 정의되어야한다. 따라서, 연산자가 많은 구성요소로 이루어진 리스트에 적용되면, 연산자는 리스트전체에 걸쳐서 일정한 적용되어야한다. 예를 들어, 3+(1 2 3)' 은 '(4 5 6)' 을 산출해낸다 (리스트의 각각의 수에 3 을 더한다).

다른 실시예로서, 'A+3'과 같은 식은 부적합한 조합으로 인식될수 있어서, 프로그램 유전자스트링이 평가될 때 상기와 같은 식은 불합격되고, 이전에 제안된 연산자로의 확장이 더 나은 방법이라고 간주된다.

입력데이터에 유전자스트링을 적용할 때는 항상 에러가 발생할 수 있다. 에러가 발생할때는 적용이 중단되고 프로그램은 매우 낮은 적합도를 갖는다. 에러는 다양한 이유로 인해 일어날수가있으나, 가장 보편적인 에러는 불가능한 동작이 시도되는 것이다. 유전자스트링은 조합으로 인해 계속적으로 변하기 때문에 '+ * 3'과 같은 부적합한 식의 발생이 가능하다. 이러한 식은 프로그램이 작동될 때 에러를 초래하므로, 프로그램스트링의 적합성이 평가되는 스텝 62 에서 낮은 적합도를 배정함으로써 제거될 수 있다.

상기 입력데이터가 유전자스트링에 적용된 후에, 본 발명에 따른 시스템은 적합함수를 이용하여 얻어진 결과를 원하는 출력과 비교하는 스텝 62 을 계속한다. 유전자스트링의 평가에 따라서 각각의 유전자스트링은 적합도값을 갖는다. 적합도값은 유전자스트링과 원하는 결과사이의 유사점을 나타낸다. 바람직한 방법으로 적합값은 숫자로서 표시된다. 적합함수는 입력데이터에 적용되는 프로그램이 얼마나 우수한지를 객관적으로 측정할수있도록 한다. 적합도의 비교는 한세대의 모든 프로그램 유전자스트링에 적용되고, 서로 관계되는 모든 유전자스트링의 서열을 이루게한다. 적합함수에 의해서 평가되어 더 높은 적합도값을 가진 프로그램 유전자스트링은 낮은 값을 갖는 다른 스트링보다 더 높은 서열에 위치한다. 초기에 임의로 생성된 대부분의 유전자스트링은 초기 선택과정의 무작위특성의 결과 때문에 낮은 적합도값을 가질 것이다. 그러나, 계속적인 유전자스트링의 생성이 이루어지면서 점점 더 높은 적합도값을 갖는 유전자스트링을 발전시킬 것이다.

시스템이 특정한 문제에 있어서 결과가 좋은지를 판단하는 능력은 갖추지 못하기 때문에 특정한 문제를 해결하기위한 적합함수는 사용자에게 의해서 제공되어야한다. 예를 들어, 손으로 쓴 글을 인식하는 시스템의 입력데이터는 유전자스트링에 적용되는 디지털화된 손으로 쓴 샘플일 수 있다. 원하는 출력은 상기 샘플에 포함된 문자의 실제스트링이다. 기대되는 문자의 스트링과 비교된 것을 생성, 출력하기위해 각 컴비네이터식은 입력데이터에 적용된다. 상기 예에서 적합도값은 정확한 위치에 배치되어 정확하게 확인된 문자의 수로 부터 결정된다. 사용자의 필요성과 기호에 따라 글자의 정확한 순서가 각각의 문자를 확인하는 것보다 더욱 중요할 수 있다. 반면, 동일한 시스템을 다른 이용자가 사용할 때, 문자의 순서보다 문자를 확인하는 것을 우선순위로 둘 수 있다.

초기의 결과는 열등하고 정확한 문자를 가지지 않는 출력을 포함할 수 있다. 유전자스트링의 평가가 문자

나 문자의 스트링을 생성하는 유전자스트링은 더 적은 문자가 확인된 다른 스트링보다 높은 적합도값을 갖는다. 프로그램의 출력이 기대되는 문자스트링을 매칭하는데 더욱 근접할수록, 더 높은 적합도값을 갖는다.

스텝 64 에서 시스템은 유전자스트링의 진화를 종결할지를 판단한다. 즉, 진화된 유전자스트링이 사용자의 목적을 충족시키기 충분한 시기를 판단하는 기준이 있어야 한다. 종결기준은 문제에 따라 달라지며, 사용자에게 의해서 공급된다. 적합함수는 하나의 해식을 다음의 식과 비교하여 우위를 판단하는데 사용된다. 전형적인 종결판단 기준은 적합함수가 적용될 때 특정 문턱값이 구해지는 것을 요구할 수 있다. 또 다른 종결기준은 프로그램 유전자스트링의 적합도가 마지막 'n' 세대동안 크게 향상되지 않았다는 것이다. 상기 종결기준은 위에서 설명한 두가지 측정을 포함할수있고, 둘중의 하나의 기준이 충족되면 진화가 종결된다.

만약 종결기준이 충족할 때, 프로그램은 종결되고 가장 우수한 프로그램 유전자스트링이 특정문제의 최선의 해결책으로서 시스템에 공급된다. 이후 상기에서 설명한바와 같이 도 5 의 스텝 96 에서 종결함수가 시스템에 입력된다.

도 4 에서 보는바와 같이, 종결기준이 도달하지 않을 때, 시스템은 동작이 프로그램 유전자스트링의 다음 세대를 형성하는데 선택되는 스텝 66 을 계속해서 실행한다. 도 5 의 스텝 92 에서 설명된 시스템파라미터로서 입력된 결합율에 의해서 유전자 결합(스텝 68) 또는 복제 (스텝 70)동작이 선택된다.

동작이 선택되면 현재의 세대에 적용되는데, 결합동작 (스텝 68)은 다음 세대의 두 개의 새로운 서로다른 유전자스트링을 생성하기위해 두 프로그램 유전자스트링을 합성하는데 관여한다. 복제동작 (스텝 70)은 현재세대에서 하나의 프로그램유전자스트링을 선택하여, 다음 세대로 복사하는 동작을 한다.

상기 두가지의 동작을 위해 현재의 세대로 부터 후보자가 선택되어야한다. 결합동작의 경우 두 개의 후보자가 선택되어야하는 반면 (스텝 72), 복제동작은 단일 후보자가 선택되어야한다 (스텝 74). 본 발명에 따른 후보자선택의 바람직한 방법은 현재 세대로 부터의 후보자의 선택에 가중치를 더하기위해 적합도의 서열을 이용하는 것이다. 상기 동작을 위해서 현 세대의 모든 적합도의 서열이 합계된다. 따라서, 후보자의 선택될 확률은 모든 후보자의 전체적합도와 비교할 때의 적합도의 비율이다.

예를 들어, 모든 유전자 스트링의 전체 적합도가 250 이고 유전자스트링 'A'의 적합도값이 25 일 때의 선택될 가능성은 $25/250=10\%$ 이다. 만약 유전자스트링 'B'의 적합도값이 12.5 일 때, 선택될 가능성은 'A'의 $1/2$, 즉 5% 이다. 우수한유전자스트링을 선택하는 다른 많은방법들이 있으나, 본 발명의 명세서에서는 상기 방법이 한 예로서 사용되었다.

결합동작에 있어서, 유전자스트링이 크로스오버 포인트로 불리는 위치에서 분열되고, 상기와 비슷하게 분열되는 다른 성공적인 유전자스트링의 부분과 결합된다. 두 유전자스트링을 결합하는 가장 간단하고 효율적인 방법은 각 유전자스트링의 임의의 포인트를 선택하여 스트링을 두 부분으로 분리한뒤에 첫 번째 유전자스트링의 각 부분을 두 번째 유전자스트링의 해당하는 부분과 결합시키는 것이다.

도 4 에서, 스텝 72 에서 두 개의 후보자 짝이 선택되면, 스텝 76 에서 크로스오버포인트는 각각의 유전자스트링마다 결정된다. 스텝 78 을 계속적으로 실행할 때, 둘 이상의 유전자스트링의 부분들이 서로 적용됨으로서 새로운 프로그램 유전자스트링을 생성한다.

스텝 79 에서 스트링의 열림괄호의 수가 같은 수의 닫힘괄호에 의해 매치된되는지를 확인하기위해 새로이 생성된 각각의 스트링이 검사된다. 상기 동작은 초기 유전자스트링이 생성되는 도 6 의 스텝 107 에서 실행되는 괄호의 균형을 맞추는 동작과 유사하다.

예를 들어 유전자스트링 S(S(B+)1C)KI 가 B(C(KS)1*)7KI와 결합될 때, 임의적 선택으로서 상기 컴비네이터 'B' 이후의 첫 번째 유전자스트링을 분할하고 컴비네이터 'I' 이후의 두 번째 유전자스트링을 분할된다. 스트링 S(S(B*)7KI 와 +)1C)KIB(C(KS)I 을 생성하기위해서 각각의 분할된 부분들을 결합시킨다. 이때, 상기 결합에서 생성된 첫 번째 유전자스트링 S(S(B*)7KI 에서는 열림괄호의 수가 닫힘괄호보다 더 많이 있으므로 닫힘괄호가 스트링의 끝에 첨가된다. 두 번째 스트링 +)1C)KIB(C(KS)I 에서는 열림괄호 앞에 두 개의 닫힘괄호가 있다. 따라서 식의 처음에 두 개의 괄호를 첨가함으로써 균형을 이룰 수 있고, 결과적으로 스트링 ((+)1C)KIB(C(KS)I 이 만들어진다. 계속해서 상기 스트링에서 식의 맨 끝에 닫힘괄호를 부가함으로써 괄호수의 균형을 이룰 수 있다. 따라서, ((+)1C)KIB(C(KS)I) 의스트링이 만들어진다.

상기 예에서는, 결합의 결과로서 부모스트링과는 다른 길이를 갖는 두 개의 유전자스트링을 생성하는 것이다. 상기 결합에 의해 생성된 프로그램 유전자스트링의 길이는 계속적인 생성에 따라 변화하므로, 유전자스트링의 추가적인 복잡성 또는 유전자스트링의 간소화를 가능하게하기 때문에, 상기의 동작은 일반적이고 필수적이다.

유전자스트링을 결합하는 여러 가지 방법이 있으나, 상기기술된 방법이 보편적으로 이용되는 방법이다. 그러나, 특정한 문제마다 서로다른 접근방법이 요구되기 때문에, 본 발명에 따른 유전자프로그램 시스템은 사용자가 결과를 무효화하거나, 결합함수 또는 선택함수와 같은 유전자프로그램의 중요한 요소들을 대체할수 있도록되어있다. 또한 시스템을 무효화할 수 있는 능력은 각 프로그램을 원하는 대로 사용하기를 원하는 사용자가 큰 통제력을 갖게한다.

스텝 70 에서, 현재 세대의 유전자스트링은 다음 세대로 복제된다. 유전자 스트링복제 동작은 단순히 유전자를 다음세대로 복사하는 것이다. 이것은 자연세계에서 개인이 다음세대에 사는것과 유사하다. 유전자스트링은 변화되는 것이 아니라 단지 다음세대를 고려하여 유전자풀로 복사되는 것이다. 다음세대로 복제되는 유전자 스트링은 일반적으로 이전세대에서 가장 높은 적합도를 갖는 스트링이다.

스텝 74 는 복제를 위한 개개의 실제선택이 이루어진다. 이러한 선택은 상기 스텝 72 에서 설명된 가중된 임의의 선택에 의해서 만들어진다.

스텝 68 또는 70 에서 스트링이 결합되거나 복제되는 경우에, 다음세대의 후보자인 스트링 모두는 각각의

스텝 80 과 82 의 치환동작 과 변환동작을 거쳐야한다.

스텝 80 의 치환동작은 도 5 의 스텝 91 에 입력되는 치환도수를 이용하고 스트링이 치환되었는지를 무작위로 검사하여 수행된다. 치환된 스트링이 있으면, 후보자유전자의 순서가 뒤바뀌게된다. 예를 들어, 스트링 S(SB(CK)SI)*7K 가 복제되고, 치환되었는지를 검사하게 된다. 검사를 통해 스트링이 다음세대에서 치환되었으면, 스트링이 S(BC(7(CK)*1)SK 또는 다른 형태의 유전자스트링 순서바뀜이 일어날 수 있다. 치환동작은 상기와 같은 스트링의 재배열을 일어나게하는 결합을위해 정확한 해석에 근접한 유전자스트링을 다른 유전자에 의지하지않고 변형하는 효율적인 방법을 제시한다. 따라서, 비교적 높은 적합도값을 가진 프로그램 유전자스트링은 다른 순서로 단순히 재배열된 유전자들의 높은 적합도값을 균일하게 갖는다.

본 발명에 따른 구현 방법에서는 치환도수에 근거하여 가능한 치환동작을 위해 스트링의 각각의 유전자를 검사한다. 만약 어떤 유전자를 치환하기로 결정하면, 다른 유전자가 무작위로 선택되고 프로그램유전자스트링안에서 상기 유전자들의 위치가 치환된다. 예를 들어, 만약 프로그램 유전자스트링 S(K*+3)C(4-)5 의 앞에 위치한 두 유전자 'S' 와 '(' 가 치환되지않고, 세 번째 유전자인 'K' 가 치환된다고 할 때, 다른 유전자 예를 들어 'C' 가 무작위로 선택되고 이 두 유전자가 치환되어 새로운 유전자스트링 S(C*+3)K(4-)5 이 생성된다. 이후 나머지유전자들도 치환동작이 끝나기전에 검사된다.

치환동작으로 인해 식의 괄호의 순서가 뒤바뀔 가능성이 있기 때문에, 스트링에서 닫힘괄호가 먼저 위치하는 경우와 같은 열림괄호와 닫힘괄호의 불균형한 순서로 인해 치환동작의 일부로서 스텝 79 에서 기술된 괄호의 균형을 맞추는 과정이 반복될 수 있다.

마지막 동작인 스텝 82 의 변환동작은 유전자가 자발적으로 다른값으로 변형되는것을 말한다. 변형동작은 도 5 의 스텝 90 에서와 같이 자신의 값을 바꾸는 주어진 유전자의 도수를 결정하는 변형도수에 근거를 둔다. 만약 유전자 스트링 S(S(B+1)C)KI 의 컴비네이터 C 에서 변형이 일어날 때, 새로운 유전자는 초기 집단을 생성하는 방법과 비슷하게 선택된다. 상기 새로운 유전자는 컴비네이터 C 를 대체하여 새로운 스트링 S(S(B+1)K)KI 을 형성하게 된다. 이러한 변형동작으로 인해 매우 다른 유전자스트링이 만들어지고, 완전히 다른 문제해결법을 생성할수있는 잠재력을 가진 새로운 유전자스트링을 형성하게된다. 따라서, 변형동작은 혁신적인 결과를 가져올 수 있다.

변형동작의 일부분으로서 괄호가 변형되었을 때, 결합을 위한 스텝 79 에서의 괄호의 균형을 맞추는 동작은 새로운 유전자스트링에 적용될 수 있다.

새로운 후보자가 치환동작과 변형동작을 위해 검사되면, 다음 세대에 포함된다. 스텝 84 에서 새로운 세대가 완성되었는가를 검사한다. 만약 완성되면, 시스템은 스텝 60 으로 되돌아가 다시 사이클을 시작하고, 그렇지않은 경우는 유전자세대를 완성하기 위해 결합 또는 복제 동작과정을 다시 시작한다.

이때, 임의로 생성된 모든 프로그램이 과정이 종결될때까지 변화없이 계속될 수는 없다는 것을 고려해야 한다. 이는 Halting 문제와 같이 컴퓨터 과학문헌에서 알려져있고, 시스템은 프로그램이 종료하는가를 예측할수없다는 것이 증명되어있다.

이것은 특정 컴비네이터 유전자스트링을 평가하는것이 출력결과를 생성하지못하는 입력데이터의 끝없는 진화를 유발하는 문제가 발생한다. 상기 문제는 평가동작의 시간을 측정하여 정해진 시간안에 결과를 생성하지 못하는 평가는 종결하여 해결될 수 있다. 주어진 시간안에 평가될 수 없는 유전자스트링은 다음세대를 위한 유전자풀의 후보에 포함되지 않거나, 매우 낮은 적합도를 갖는다.

상기 문제의 해결을 위한 다른 접근방법은 도 3 에서 도시된바와 같은 병렬프로세스 시스템에서 가능하다. 이 경우 독립된 프로세서 (또는 상기 시스템이 소프트웨어장치로서 구현될때의 과정)가 유전자풀의 각각의 프로그램 유전자스트링을 평가한다. 유전자스트링들의 평가가 끝났을 때, 상기 스트링들은 상기 스텝 68 과 70 에서 기술된 복제와 결합동작을 위한 후보자가 된다. 초기에는 후보자스트링이 조금 밖에 되지않으나, 이후 후보자풀의 크기가 증가한다. 낮은 적합도값을 가진 프로그램 유전자스트링은 더 높은 값을 갖는 유전자스트링에 비해 매력적이지 못하다. 더 높은 적합도값을 가진 유전자스트링은 이후에 유전자풀에 입력될 더욱 매력적인 유전자스트링을 기다릴 때 낮은 적합도를 가진 유전자스트링과의 결합을 저지한다. 그러나, 일정한 기간이 지난뒤에 적합한 유전자스트링이 나타나지 않는다면 높은 적합도값을 가진 유전자스트링은 낮은 적합도를 가진 유전자스트링과 결합할 수 있다.

상기와 같은 해결법은 후보자 유전자에서 나타나는 유전자스트링 전부를 기다리지 않음으로서 위에서 언급된 Halting 문제를 피할 수 있다. 본질적으로 유전자스트링이 후보자풀에 들어가는데 오랜 시간이 걸린다면, 이것은 결합동작에 이용할수 없고, 따라서 상기 유전자스트링이 매우 뛰어난 적합도를 갖지않는 이상, 다음 세대로 계속 이어질수없다. 출현하지않는 유전자스트링은 결코 결합할수없다. 이는 결합유효성이 본질적으로 추가된 적합도 측정기준인 자연적선택과 유사하다.

상기와 같은 시스템에서는, 너무 많은 프로세서들이 끝나지 않는 문제를 계속 평가하는데 관련되는 상황을 피하기위해 추가적인 검사가 반드시 실행되어야한다. 만약 대부분 또는 모든 컴비네이터장치가 무한의 유전자스트링을 평가한다면, 유전자 프로그램시스템은 비효율적으로 운영되고, 결코 문제를 해결하지 못할 수 있다. 이러한 상황은 유전자스트링의 평가에 시간제한을 둠으로서 피할 수 있다. 만약 컴비네이터 장치가 주어진 시간안에 특정 유전자스트링을 평가할수없을 때, 평가는 중단되고 새로운 유전자스트링이 프로세서에 정해진다. 예를 들어, 주어진 프로그램 유전자스트링을 평가하는데 5초로 시간을 제한할 수 있다. 만약 5초안에 평가가 끝나지않으면, 평가는 중단되고 평가중이던 유전자스트링은 버려진다. 따라서, 5초안에 평가되지 않은 유전자스트링은 다음세대에 살아남지 못할 것이다. 평가의 시간제한은 유전자스트링의 복잡성에 따라서 다르게 정해질 수 있다.

프로그램 유전자스트링이 그 자신을 완전히 해결하지 못할 때, 비슷하지만 덜 복잡한 문제가 발생한다. 즉, 평가가 끝난뒤에 결과는 여전히 컴비네이터와 연산자를 포함한다. 예를 들어, 입력 데이터 3 이 적용될 때, 프로그램 유전자스트링 * S * 1 는 * 9 의 식으로 풀릴 수 있는 * * 3 3 의 식을 생성할 수 있다. 유전자 * 의 개체정의가 규정되지 않았다면, 상기 유전자 * 를 위한 두 번째 요소를 필요로하는 상기 식은 불완전하다.

상기 문제를 처리하기 위한 가장 간단한 방법은 상기 식에 낮은 적합도를 배정하는 것이다. 그러나, 바람직한 실행방법은 입력데이터에 상기 식을 다시 적용하는 것이다. 상기 예에서 27의 결과를 산출하는 식 * 9 3을 생성하기 위해 식 * 9는 입력데이터 3에 적용된다.

이전에 기술되었듯이, 도 2는 컴비네이터를 포함하는 유전자스트링을 평가하는 컴비네이터장치를 설명하고 있다. 컴비네이터장치는 트리포맷의 구성요소들을 저장하는 주 메모리 (42)와 초기컴비네이터로서 구현되는 컴비네이터를 포함한다. 이러한 장치는 효율적으로 작동하고 ASIC와 같은 단일 주문집적회로로서 구현이 가능하다. 또한 다른 컴비네이터장치와 함께 작동할 수 있다.

컴비네이터장치는 컴비네이터를 스택조정으로 구현함으로써 소프트웨어에서 구현될 수 있다. 이때, 컴비네이터가 작용하는 트리는 현재 사용되는 대부분의 컴퓨터에서 이용가능한 푸시다운 스택기술에 설정될 수 있다. 평가트리는 장치의 스택의 일련의 지시자로서 사용되고, 컴비네이터는 적절하게 스택에서 구성요소들끼리 순서를 재배열하여 트리를 변경한다. 현재 사용되는 대부분의 프로세서가 효율적인 스택조정 연산자를 가지지만 트리구조를 조정하는데에는 덜 효율적인 연산자를 가지기 때문에 상기 동작은 효과가 있다.

컴비네이터를 가진 유전자스트링을 나타내는것외에, 컴비네이터가 보편적인 장치를 제공하기 때문에 유전자알고리즘 연산자는 컴비네이터의 형태로 표현될 수 있다. 따라서, 만약 전문화된 컴비네이터장치가 구성되면 컴비네이터식 이외의 다른 것을 인식하는 장치를 만들필요는 없다.

위에서 설명한바와 같이 도 3은 병렬연결된 복수의 컴비네이터장치를 갖는 유전자 프로그램시스템을 도시하고 있다. 각 세대마다 계속적으로 평가되어야 하는 유전자들의 여러 유전자스트링을 갖는 필요성 때문에, 병렬 프로세스시스템을 구성하는 더욱 수월하고 효과적인 방법이 있다. 도 3에서와 같이, 각각의 컴비네이터장치는 입력데이터가 주어진 컴비네이터 유전자스트링을 평가할 수 있다. 이러한 컴비네이터장치는 유전자들의 각각의 유전자스트링을 평가하는데 이용된다. 따라서, 유전자들의 각각의 유전자 프로그램스트링의 순차적평가보다는 모든 유전자스트링이 동시에 평가되어 효율성과 속도를 향상시킨다.

도 3의 GP제어부 (52)는 유전자 프로그램시스템전체를 제어하고, 컴비네이터장치 (50)를 관리하기 위해 데이터/제어버스 (46)를 이용한다. 상기 GP제어부 (52)는 시스템을 진행하는 일부분으로서 구성되는 GP메모리 (54)에 저장되어 있는 입력 기준, 데이터, 파라미터를 사용한다. 상기 GP메모리 (54)는 스트링의 길이, 변형, 결합율, 또한 입력데이터, 적합함수, 종결기준과 같은 정보를 저장한다. 이때, 종결기준은 진화된 프로그램 유전자스트링의 유효성을 판단한다.

컴비네이터장치 (50)는 입력데이터를 받은 임의의 유전자스트링을 평가할 수 있고, 또한 유전자알고리즘에서의 필요한 동작을 실행하기 위한 명령을 저장한다.

초기의 GP제어부 (52)는 컴비네이터, 상수, 연산자로 부터 임의의 유전자스트링을 생성하기 위해 각각의 컴비네이터장치 (50)를 관리한다. GP제어부 (52)는 데이터/제어버스 (46)를 따라 전달되는 생성결과를 분석하고, 각각의 유전자스트링에 대해서 이후에 어떤기능을 수행해야 하는가에 관한 적절한 명령을 각각의 컴비네이터장치에 내림으로서 유전자의 결합, 치환, 그리고 다른 유전자의 기능을 제어한다.

이러한 방법을 바탕으로, 다른 컴비네이터장치의 유전자스트링과 결합하거나, 재생성되는 결과에 따라서 유전자스트링이 컴비네이터장치를 사이에서 전달된다. 이러한 유전자스트링이나 유전자스트링의 단편들은 프로그램 유전자스트링 버스 (48)를 따라 이동한다.

도 3에서 도시된 병렬시스템은 더 우수한 해결책을 위해 유전자 진화과정을 매우 가속화한다. 이것은 유전자스트링을 순차적으로 평가하는 것이 아닌 모든 프로그램 유전자스트링이 동시에 평가될수있기 때문이다. 컴비네이터장치 (50)는 비교적 저렴하고 설치가 간단하다. 따라서, 크고 대규모로 병렬연결된 연산시스템에서는 수백 또는 수천 컴비네이터장치를 이용하여 구성될 수 있다.

긴 프로그램 유전자스트링이나 컴비네이터장치가 사용되지않는 문제, 즉 사용되는 컴비네이터장치보다 유전자들의 후보자가 더 적을 때, 평가되는 단일 유전자스트링이 여러 부분으로 나누어져서 여러 컴비네이터장치가 각각의 나누어진 유전자스트링 부분들을 평가한다. 모든 컴비네이터가 순서와 관계없이 평가될 수있고 모든 컴비네이터부분들이 다시 합하여져도 같은 결과를 유지하기 때문에, 상기 동작이 가능하다. 즉, 단일 유전자스트링은 서브스트링 A, B, C로 나누어질수있으며 각각의 서브스트링은 독립적으로 평가된다. 다시 합쳐진 서브스트링의 결과는 단일스트링으로 평가된 결과와 동일한값을 갖는다.

[순회 세일즈맨의 실시에]

본 발명을 잘 알려진 순회 세일즈맨문제에 적용함으로써 본 발명에 따른 일 예를 설명한다. 상기 예는 목적이 세일즈맨이 방문장소를 순회하는데 최단루트를 찾는 것을 최적화하는 문제이다. 방문장소 사이의 거리의 도표는 입력데이터로서 공급되고, 원하는 루트는 적어도 한 번은 각 장소를 방문하는 동안의 최단루트이다.

이때, 최단거리를 제공하는 알고리즘이나 컴퓨터프로그램은 없다. 많은 선택

의 수 때문에 ($n-1$ 계승에서 n 은 상기 문제에서 도시의 수), 모든 실현가능한 루트를 계산하는 간단한 방법은 매우 비효율적이다. 예를 들어, 15개의 도시를 방문해야하는 상황에서는 거의 90,000,000,000개의 각각의 도시를 방문하는 루트가 가능하다.

본 발명에서는 도시사이의 거리가 주어진 더 짧은 루트중의 하나를 생성하는 문제 해결법을 찾을 수 있다. 먼저, 사용자는 입력데이터, 즉 방문해야할 도시들 사이의 거리를 포함하는 트리구조를 만든 뒤, 적합함수와 종결함수를 입력한다.

상기 문제에 접근하기 위한 두가지 방법이 있다. 첫 번째 방법은 프로그램 유전자스트링으로 기록되는 알려진 방법이나 해결법을 이용하는 유전자프로그램의 약한형태 (weak form)라고 불리는 방법이다. 이러한 알려진 프로그램은 초기 유전자스트링풀을 모집단하는데 쓰인다. 이는 더 나은 해결법을 찾기위한 시도에서 양호한 초기유전자스트링으로 시작하는 경우이다. 이러한 방법은 모든 초기 유전자스트링이 알려진 해

결법을 나타내어 혁신적인 결과를 생성할 것 가능성이 적기 때문에, 유전자프로그램의 약한형태라고 한다. 알려진 해결법은 이미 비교적 만족스러운 결과를 생성하기 때문에, 유전자 프로그램시스템이 종래의 방법과는 전혀 다른 혁신적인결과를 생성할것지는 않다.

그러나, 순회 세일즈맨문제를 해결하기위해서 유전자프로그램의 강한형태 (strong form)가 사용된다. 이 방법은 적합성에 따라 평가되고 선택되는 임의의 생성된 프로그램 유전자스트링을 가진 초기 유전자스트링풀을 모집단한다. 비록 상기 유전자프로그래밍의 강한형태가 열등한결과를 생성하고 우수한 유전자스트링을 구하는데 오랜시간이 걸릴지라도, 혁신적으로 새롭고 잠재적으로 우수한 프로그램해결방법을 생성하는 가능성을 제공한다. 원래 초기 프로그램 유전자스트링은 사용자가 문제에 대한 좋은 해결법이라고 믿는 어떤 예견된 생각도 포함하지않는다. 대신, 유전자 프로그래밍시스템은 자신을 해결법의 한 부분으로 제한하기 보다는 문제를 해결하는 가능한 모든 방법을 평가한다.

순회 세일즈맨문제를 해결하기 위해, 먼저 도시간 거리표의 입력데이터를 포함하는 트리구조(또는 수학적 그래프)가 만들어진다. 컴비네이터프로그램이 그래프상태로 작용하기 때문에, 입력데이터는 반드시 그래프로 나타난다. 이러한 구조는 도 7 에 도시되어있다. 도 7 은 각각의 가지가 서브트리인 트리구조를 도시하고 있다. 이때, 서브트리는 가지로 표현된 도시와 트리구조의 다른 도시들과의 거리를 나타낸다. 도 7 의 하단에 표시된 문자스트링은 동일한 그래프의 스트링표시이다. 이것은 동일한 구조를 기술하는 더욱 간결한 방법이다. 예를 들어, 트리의 첫 번째 가지는 도시 'A' 와 다른 도시들의 거리를 나타낸다. 각각의 가지는 거리를 나타내는 작은가지와 도시를 나타내는 라벨가지를 포함한다. 따라서, 'A' 에서 'B' 의 거리는 12, 'A' 에서 'C' 는 17, 'A' 에서 'D' 의 거리는 45 이다. 이와 유사한 방법으로 다음으로 위에 위치한 가지는 도시 'B' 에서 다른 도시들과의 거리를 나타낸다. 즉 B 에서 A 는 12, B 에서 C 는 27, B 에서 D 는 32 이다. 상기 트리구조는 도시 'D' 와 다른 도시들의 거리를 나타내는 마지막 서브트리까지 이어진다.

본 발명에 의해 생성된 프로그램 유전자스트링은 결과를 생성하기 위해 이러한 입력트리에 적용된다. 원하는 결과는 방문되는 도시의 순서에 따른 도시의 리스트를 구하는 것이다.

적합함수는 최단거리를 생성하는 프로그램 유전자스트링에 가장 높은 평가등급을 내려야한다. 그러나, 적합함수는 완성루트를 생성하지않은 후보자들 또한 평가해야한다. 따라서, 특정 프로그램 유전자스트링이 도시의 완전한 리스트를 선택하는가를 적합도의 첫 번째 기준이어야 한다.

적합도는 순회되는 총 거리에 달려있기 때문에, 바람직한 적합함수는 각각의 프로그램의 유전자스트링마다 서로다른 값을 만들고, 이때 더 낮은 값을 가질수록 더 우수한 유전자스트링이다. 따라서, 짧은 순회거리를 생성하는 프로그램의 유전자스트링은 높은 적합도값을 가질 것이다. 반대로, 긴 순회거리를 생성하는 유전자스트링은 낮은 적합도값을 가질 것이다.

해결법의 효율성을 평가위해서, 시험적인 해결법은 최악의 경우에 일어날 수 있는 해결법과 비교된다. 최악의 경우를 고려한 해결법은 루트에서 두 도시사이의 가장 긴 거리를 구하고, 이 거리를 도시의 수로 곱하여 얻을 수 있다. 예를 들어, 도 7 에서 4 개의 도시들중 최장거리는 A 와 D 사이의 거리인 45 이고, 이 거리를 4 로 곱하여 최대 거리인 180 의 결과를 얻을 수 있다 ($4 \times 45 = 180$).

상기 해결법은 각도시들의 거리가 동일하지않은한 (즉, 모든도시가 서로 같은 거리에 위치하는 도 7 에서 도시된 문제의 경우가 아닌) 모든 도시를 방문하는 루트가 가장 긴 거리를 만들수없기 때문에 여느 다른 해결법보다 큰 결과를 가진다. 결과적으로, 상기 이러한 적합함수는 루트들의 객관적평가를 하는데 이용된다. 이는 제한된 루트의 순회거리를 최악의 경우를 고려한 루트로 나눔으로서 순회의 효율비를 생성하여 얻을 수 있다.

적합도의 기준은 다른 함수식에 의해서 결정된다:

$$f = (w(n-v)+d)/w$$

이때, w 는 최악의 경우를 고려한 거리 (180), n 은 전체루트에 있는 도시의 수 (4), v 는 제한된 루트에서 방문된 도시의 수, 마지막으로 d 는 제안된 루트로 순회된 거리를 나타낸다.

따라서, 도 7 의 문제해결을 위한 적합함수는 $f = (180(4-v)+d)/180$ 이다.

종결되지 않는 프로그램은 최악의 경우를 고려한 거리를 제한된 루트에서 빠진 도시들의 수를 곱하는 형태로 큰 벌점을 갖기 때문에, 상기 적합함수는 루트를 끝내는 데에 프리미엄을 붙인다. 반면, 세일즈맨의 영역에 있는 모든 도시를 방문하도록 제한된 모든 루트에는, 식에서 $n-v$ 의 결과는 제로가 되므로 함수의 처음절반이 0 으로 감소하므로, 두 번째 부분인 d/w 가 결정적 요인으로 작용할 것이다. w가 상수이기 때문에, d 의 값이 작을수록 효율비가 더 작게 나타나고 적합도는 더욱 높게 나타난다. 거리가 비율로 표시되기 때문에, 루트의 비례함수는 거리를 측정하는 단위에 영향을 받지않을 것이다.

적합함수는 킬로미터, 마일, 피트 등의 단위와 함께 잘 작용할 수 있다. 그러나, 정확한 결과를 생성하기 위해 동일한 측정단위를 사용하여 모든 거리의 값이 표현되어야한다. 즉, 모든 측정치수는 마일 또는 킬로미터 등의 하나의 단위로 통일되어야한다.

마지막으로 특정 프로그램 유전자스트링이 양호할때를 결정하는 종결기준은 사용자에게 의해서 제공되어야한다. 이러한 경우는 우리는 결과에서 부족한 점을 간단하게 찾을 수 있다. 만약 100 세대에 있어서 (한 세대는 도 4 의 스텝 60 에서 84 의 실행으로 생성된다), 최적후보자는 5% 이상 향상되지 않으며 진화가 끝나고 최적의 적합도를 갖는 프로그램의 유전자스트링은 문제의 해결법으로 사용된다.

그러나, 우리는 특정 데이터에 맞는 최적 프로그램의 해결법을 구하기 때문에, 이 과정은 서로 다른 문제를 위해 반복적으로 수행된다. 이는 적합도를 검사하는 많은 상황에서 살아남은 자연적으로 진화한 생물과 유사하다. 바꾸어 말하면, 프로그램의 해결법을 생성하는 입력데이터의 세트가 전반적 해결법을 생성하기에 충분히 다양하지않기 때문에, 특정 입력 데이터와 우수하게 작용하는 해결법이라도 다른 데이터에는 적용이 안될 수 있다.

이러한 경우, 초기데이터로 부터의 최선의 해결법은 다른 입력데이터와 비교되어 검사된다. 이 과정은 연속적으로 미세한 차이를 생성하여 많은 문제를 위한 우수한 결과를 생성하는 해결법을 생성하기 위해 계속적으로 반복된다.

상기 순회 세일즈맨 문제의 예는 간단히 설명의 목적으로 4 개의 도시로 제한되어있다. 그러나, 동일한 트리구조는 어떠한 수로 구성된 도시의 거리를 나타내는데 사용될 수 있다. 도시들이 많이 있을때는 단순히 입력데이터를 나타낼 큰 트리구조를 사용하면된다. 동일한 적합함수 $[f=(w(n-v)+d)]$ 는 도시의 수에 관계없이 이용될 수 있다 (최악의 경우를 고려한 거리 w 와 전체루트의 도시의 수 n 가 상황에 적절하게 바꾸어 적용한다). 도시의 수가 증가할수록, 해결법을 생성하는데 필요한 시간이 증가된다. 그러나, 프로그램의 유전자스트링을 생성, 진화, 평가하는 과정은 변화없이 그대로 유지된다. 따라서, 유전자 프로그래밍시스템은 특정형태의 문제를 해결하기위해 발달되기 때문에, 동일한 프로그래밍시스템이 다양한 입력데이터와 함께 주어진 문제를 해결하기위한 최적의 프로그램을 결정하는데 사용될 수 있다.

상기 발명이 전술된 실시예에 대해 주로 기술되었지만, 본 발명은 반드시 이런 실시예로 제한되지 않는다. 따라서, 여기에서 기술되지 않은 다른 실시예 변형 및 증진에 대한 것은 반드시 본 발명의 범주로부터 배제되지 않고 아래의 첨부된 청구항의 범위에 의해 한정된다.

(57) 청구의 범위

청구항 1

첫째 해결법과 둘째 해결법의 비교우위를 측정하는 적합함수를 정의하는 첫 번째 단계;

해결될 문제로부터 입력데이터를 결정하는 두 번째 단계;

각각의 프로그램 유전자스트링이 문제의 잠재적인 해결법을 의미하고, 그래프감소 연산자를 포함하는 복수의 프로그램 유전자스트링을 생성하는 세 번째 단계;

각각의 프로그램 유전자스트링의 해결법을 생성하기위해 상기 입력데이터를 각각의 프로그램 유전자스트링에 적용하는 네 번째 단계;

프로그램 유전자스트링 해결법과 적합함수를 비교하여 각 프로그램 유전자스트링을 평가하는 다섯 번째 단계;

적합도의 평가에 따라서 상기 프로그램 유전자스트링을 진화하는 여섯 번째 단계;

종결기준이 충족할 때까지 네 번째 에서 여섯 번째 단계가 반복되는 일곱 번째단계를 포함하는 것을 특징으로하는 유전자 프로그래밍기술을 이용하여 문제를 해결하기 위해 컴퓨터에 구현된방법.

청구항 2

제 1 항에 있어서, 상기 유전자스트링을 진화하는 여섯 번째 단계는 적어도 복제, 치환, 변형, 결합중의 하나를 포함하는 것을 특징으로하는 유전자 프로그래밍기술을 이용하여 문제를 해결하기 위해 컴퓨터에 구현된방법.

청구항 3

프로그래밍문제를 해결하기위한 유전자 프로그래밍시스템에서,

유전자 프로그래밍시스템의 동작을 제어하는 유전자프로그램 제어기;

상기 유전자 프로그래밍시스템의 동작과 연관되어 데이터를 저장하고 유전자프로그램 제어기와 전기적으로 연결된 유전자 프로그램메모리부;

유전자프로그램 제어기와 전기적으로 연결되어 데이터를 유전자 프로그래밍시스템으로 입력하는 입력장치;

상기 유전자프로그램 제어기와 전기적으로 연결된 그래프감소 장치;

상기 유전자 프로그래밍시스템에 저장되고, 원하는 해결법에 접근하도록 진화되고, 그래프 감소연산자를 포함하는 복수의 프로그램 유전자스트링으로 구성되는 것을 특징으로하는 전기회로의 유전자 프로그래밍 시스템.

청구항 4

임의의 입력이 주어진 문제해결을 위해 프로그램을 구하는 유전자 프로그래밍시스템에 있어서,

다양한 정보를 저장하는 메모리장치;

문제해결법을 반드시 갖는 정보를 포함하는 입력데이터를 입력하는 상기 메모리장치에 저장된 입력장치;

첫 번째 해결법과 두 번째 해결법의 비교우위를 판단하는 기준을 제공하고 상기 메모리장치에 저장된 적합함수;

메모리장치와 전기적으로 연결된 중앙처리장치;

상기 유전자 프로그래밍시스템에 저장되고, 한 개이상의 그래프감소 연산자를 갖는 복수의 프로그램 유전자스트링을 포함하는 것을 특징으로하는 유전자 프로그래밍시스템.

청구항 5

제 4 항에 있어서, 상기 중앙처리장치는 콤비네이터 평가부를 포함하는 것을 특징으로 하는 유전자 프

그래밍시스템.

청구항 6

서로 전기적으로 연결된 다수의 중앙처리부와 중앙처리부와 전기적으로 연결된 입력 장치로 구성된 유전자 프로그래밍시스템있어서, 임의의 입력이 주어진 문제를 해결하는 최적의 프로그램을 구하는 방법과 첫째 해결법과 둘째 해결법의 비교우위를 측정하는 방법은,

첫째 해결법과 둘째 해결법의 비교우위를 측정하는 적합함수를 정의하고;

해결될 문제로부터 입력파라미터를 결정하고;

각각의 프로그램 유전자스트링이 문제의 잠재적인 해결법을 의미하고, 그래프감소 연산자를 포함하는 복수의 프로그램 유전자스트링을 생성하고;

각각의 프로그램 유전자스트링의 해결법을 생성하기위해 상기 입력파라미터를 각각의 프로그램 유전자스트링에 적용하고;

프로그램 유전자스트링 해결법과 적합함수를 비교하여 각 프로그램 유전자스트링을 평가하고;

적합도의 평가에 따라서 상기 프로그램 유전자스트링을 진화하고;

종결기준이 충족할 때까지 상기 단계를 반복되는 과정을 포함하는 것을 특징으로하는 임의의 입력이 주어진 문제를 해결하는 최적의 프로그램을 구하는 방법과 첫째 해결법과 둘째 해결법의 비교우위를 측정하는 방법.

청구항 7

정의된 입력을 갖는 프로그래밍문제의 해결법을 구하는 유전자 프로그래밍시스템과 첫째 해결법과 둘째 해결법의 비교우위를 판단하는 방법에 있어서,

상기 프로그래밍시스템에 입력데이터를 저장하는 수단;

상기 시스템에 적합함수를 저장하는 수단;

상기 시스템에 종결기준을 저장하는 수단;

그래프감소 연산자로 이루어진 복수의 유전자를 저장하는 수단;

복수의 프로그램 유전자스트링을 생성하기위해 상기 유전자에 반응하는 수단;

출력을 생성하기위해 상기 입력데이터를 각 프로그램 유전자스트링에 적용하는 수단;

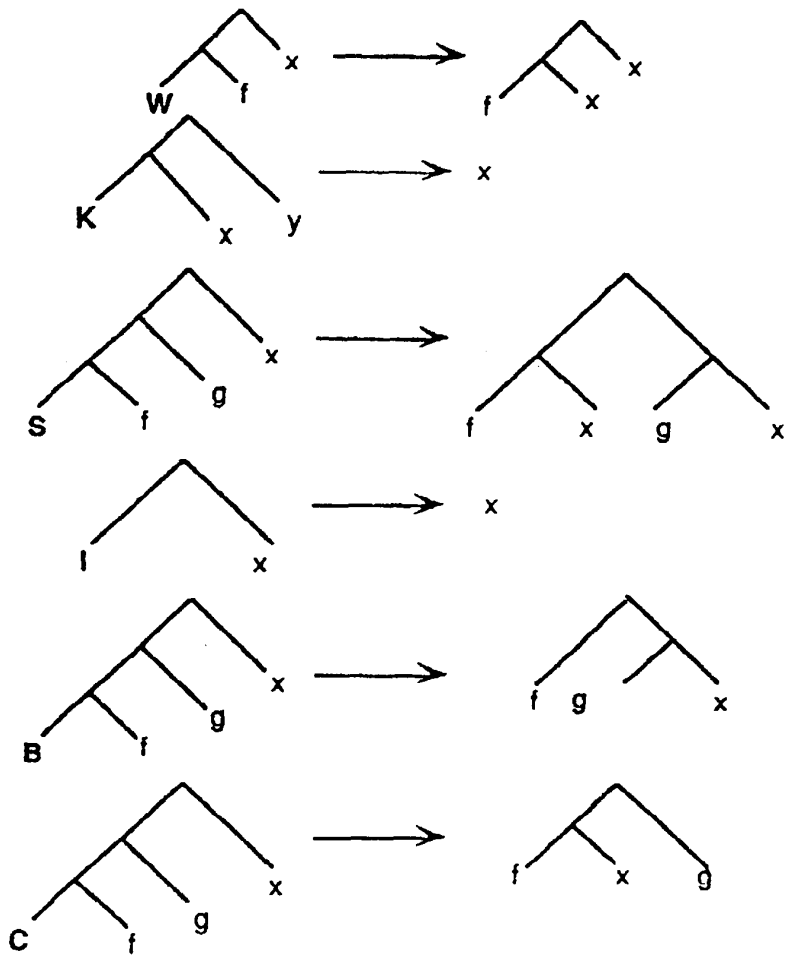
각 프로그램 유전자스트링의 적합도를 판단하기위해 상기 적합함수로서 상기 출력을 평가하는 수단;

프로그램 유전자스트링을 진화하기위해 각 프로그램 유전자스트링의 적합도에 반응하는 수단;

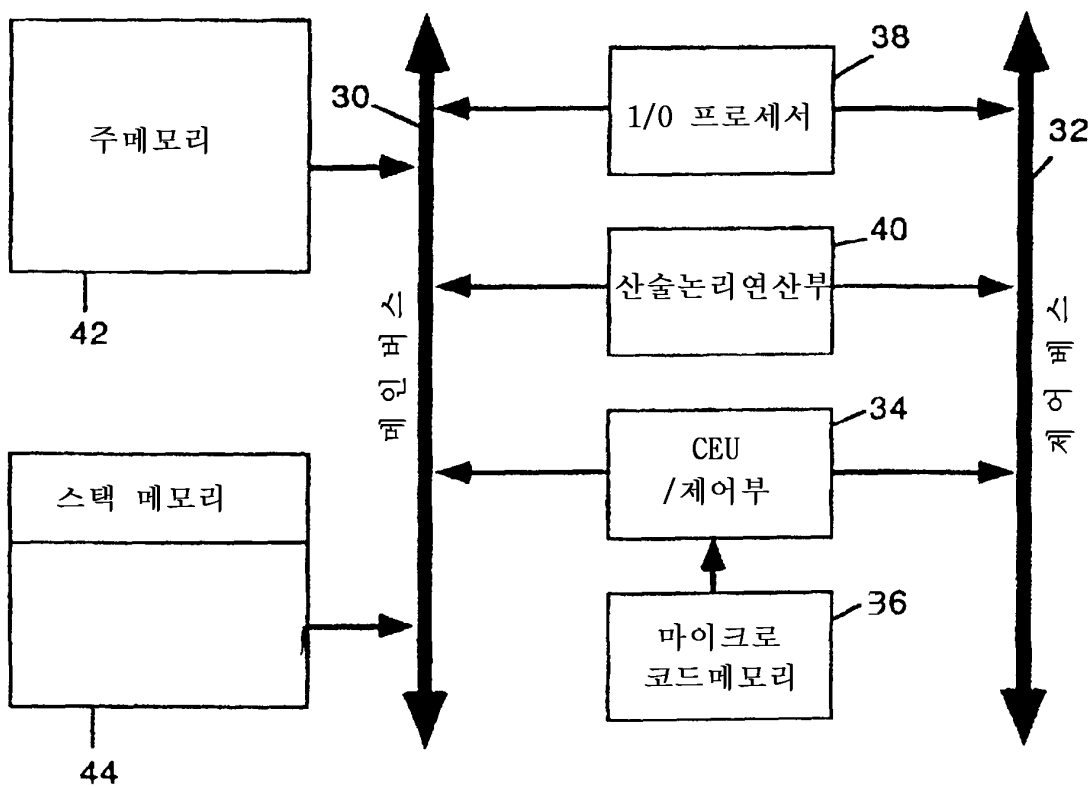
상기 종결 기준이 충족될때까지 상기 프로그램 유전자스트링을 반복적으로 진화시키고 평가하는 수단을 포함하는 것을 특징으로 하는 유전자 프로그래밍시스템.

도면

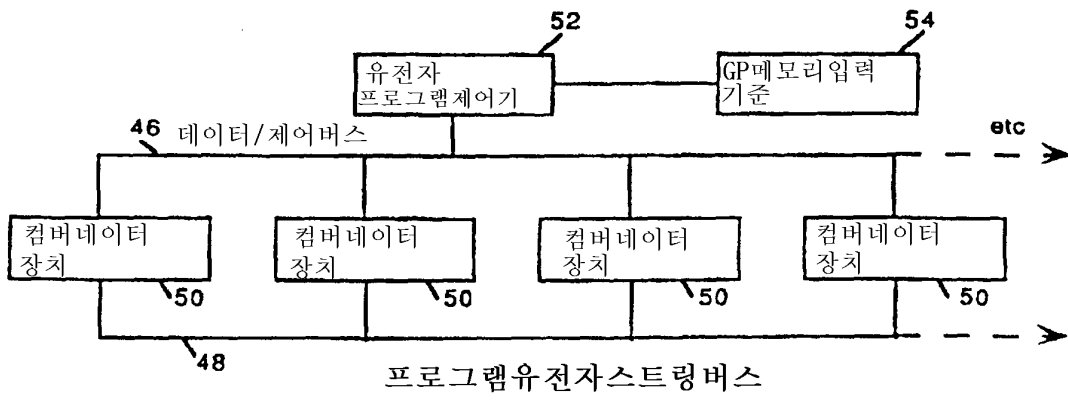
도면1



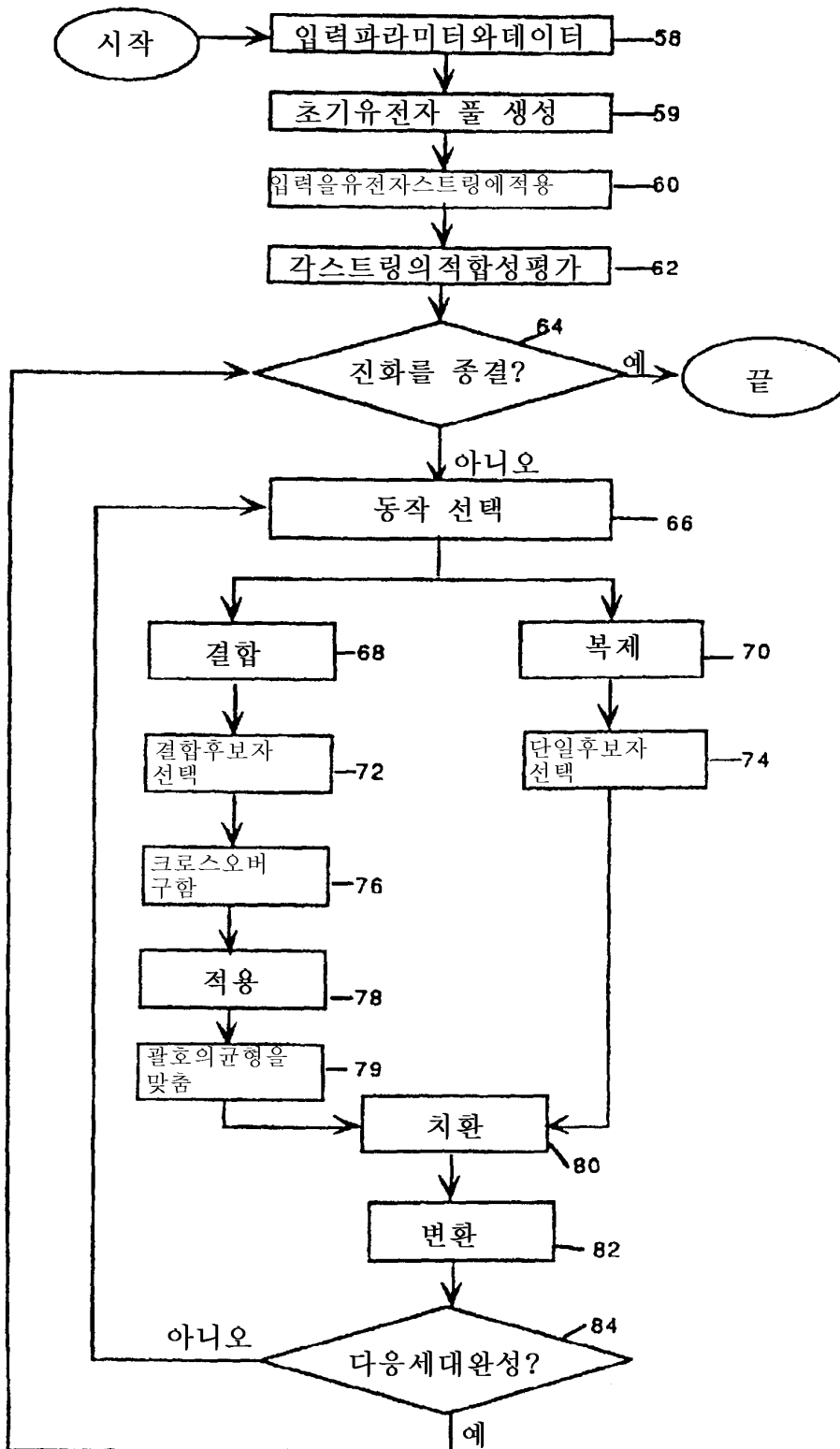
도면2



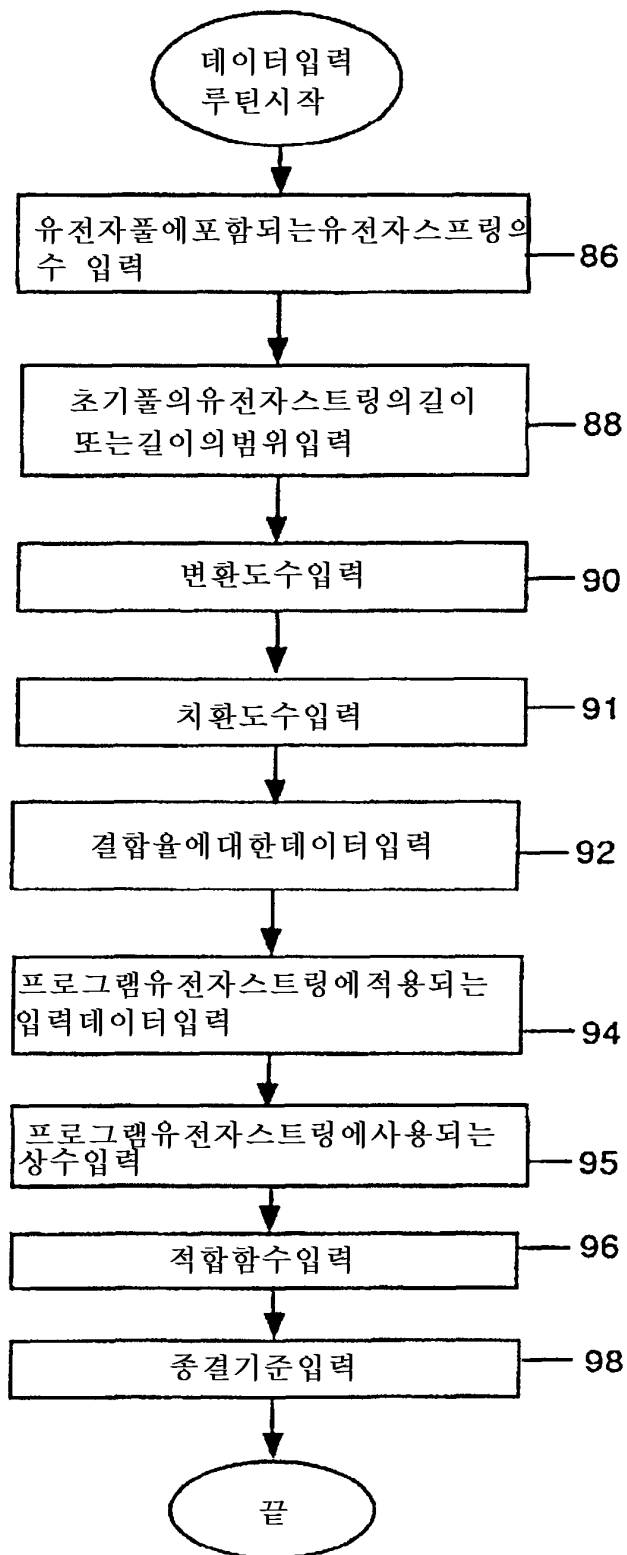
도면3



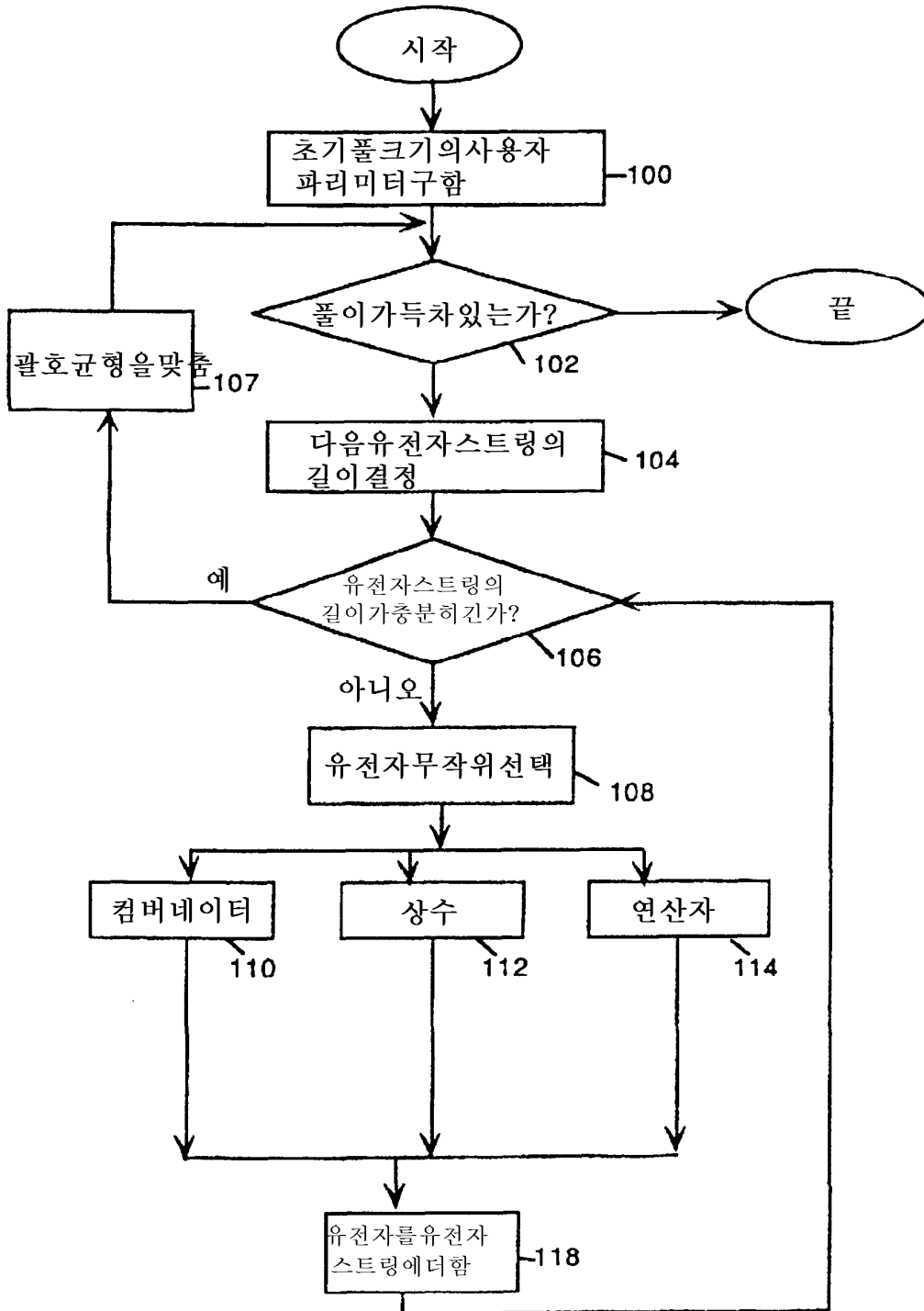
도면4



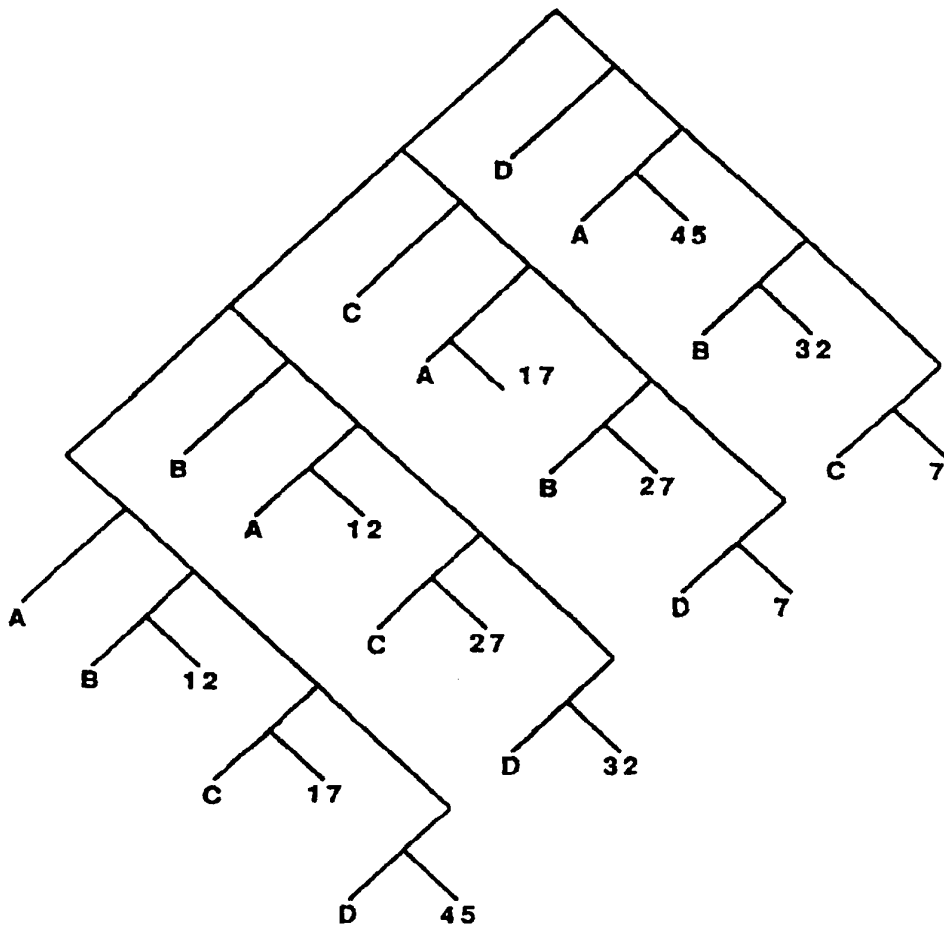
도면5



도면6



도면7



```
( (A(B 12) (C 17) (D 45))
  (B(A 12) (C 27) (D 32))
  (C(A 17) (B 27) (D 7))
  (D(A 45) (B 32) (C 7)))
```

도면8

