유전자 알고리즘을 이용한 OPKFDD의 최적화

(Optimization of OPKFDDs using Genetic Algorithms)

정 미 경 † 신 윤 정 †† 이 귀 상 †††
(Migyung Jeong)(Yoon Jeong Shin)(Gueesang Lee)
장 준 영 †††† 배 영 환 ††††† 조 한 진 †††††
(June Young Chang)(Young Hwan Bae)(Hanjin Cho)

요 약 OPKFDD(Ordered Pseudo-Kronecker Functional Decision Diagram)는 각 노드에서 다양한 확장방법(decomposition)을 취할 수 있는 Ordered-DD(Decision Diagram)의 한 종류로서 각 노드마다 Shannon, positive Davio, negative Davio 확장중의 하나를 사용하도록 하며 다른 종류의 DD와 비교해서 작은 수의 노드로 함수를 표현할 수 있다. 그러나 각 노드마다 각기 다른 확장 방법을 선택할 수 있는 특징 때문에 입력 노드에 대한 확장 방법과 입력 변수 순서의 결정에 의해서 OPKFDD의 크기가 좌우되며 최소의 노드 수를 갖는 OPKFDD의 구성은 매우 어려운 문제로 알려져 있다. 즉, OPKFDD에서의 입력변수와 각 노드의 확장 방법을 병행해서 최적의 해를 구하기 위해서는 n 개의 입력변수에 대해서 $n! \cdot 3^{2^n-1}$의 경우의 수를 고려해야 한다. 따라서 본 논문에서는 주어진 불리안 함수를 OPKFDD의 최적화 표현을 위해 노드 수를 기준으로 하여 입력변수 순서와 각 노드의 확장 방법을 함께 고려하는 혼용 유전자 알고리즘을 제안하며 최소의 노드 수를 갖는 OPKFDD를 생성하기 위해서 다양한 파라미터 값에 따른 실험결과를 제시한다.

Abstract OPKFDD(Ordered Pseudo-Kronecker Functional Decision Diagram) is one of ordered-DDs(Decision Diagrams) in which each node can take one of three decomposition types: Shannon, positive Davio and negative Davio decompositions. Whereas OBDD(Ordered Binary Decision Diagram) uses only the Shannon decomposition in each node, OPKFDD uses the three decompositions and generates representations of functions with smaller number of nodes than other DDs. However, this leads to the extreme difficulty of getting an optimal solution for the minimization of OPKFDD. Since an appropriate decomposition type has to be chosen for each node, the size of the representation is decided by the selection of the decomposition type as well as the variable ordering of the diagram. In an exhaustive method, $n! \cdot 3^{2^n-1}$ cases should be considered to get an optimal solution, where n is the number of input variables. To get an optimal solution in such a large solution space, this paper presents a hybrid genetic algorithm for the optimization of OPKFDD and experimental results are given with various parameter values.

† 정 회 원 : 전남대학교 전산학과
mgjung@chonnam.ac.kr
†† 정 회 원 : 광주대학교 컴퓨터전지통신공학부 교수
syj@honey.kwangju.ac.kr
††† 종신회원 : 전남대학교 전산학과·정보통신연구소 교수
gslee@chonnam.chonnam.ac.kr
†††† 종신회원 : 한국전자통신연구원 집적회로설계연구부 연구원
jychang@etri.re.kr
††††† 비 회 원 : 한국전자통신연구원 집적회로설계연구부 연구원
yhbae@etri.re.kr
hjcho@etri.re.kr
논문접수 : 2000년 2월 1일
심사완료 : 2000년 9월 6일

1. 서 론

최근에 불리안 함수로 대변되는 논리회로의 표현을 위해 그래프를 이용한 표현 방법인 DD(Decision Diagrams)가 제시되어 논리합성, 테스트 그리고 회로검증(verification)과 같은 여러 가지 설계자동화문제에 활발히 적용되어 왔다. 여러 가지 종류의 DD중에서 실제로 가장 많이 사용하고 있는 것은 OBDD(Ordered Binary Decision Diagrams)이다[1]. 현재 OBDD의 응용범위는 더욱 넓어져가고 있으며 이는 불리안 함수 외에도

여러 가지 이산함수(discrete function) 즉, 다치함수, 큐브 집합(cube set), 산술함수(arithmetic function)등의 경우에도 적용되며 이와 같은 논리함수의 그래프표현은 전산, 전자분야뿐 아니라 다른 많은 분야에서도 매우 유용하게 쓰이고 있다. 또한 장기적으로 이러한 그래프 표현은 그 특성인 정규적(canonic) 표현임과 동시에 사용의 편리함 등으로 인하여 더욱 그 사용이 늘어날 전망이다.

DD의 사용이 활발해짐에 따라 여러 가지 형태의 DD가 제시되었으며 그 중에서도 두드러진 것은 OFDD (Ordered Functional Decision Diagram)이며[2] 이는 같은 변수를 나타내는, 즉 같은 level의 노드에서 positive Davio 와 negaive Davio중의 하나를 사용하도록 한다. OFDD로부터 OKFDD(Ordered Kronecker Functional Decision Diagram)이 제안되었으며 이는 같은 레벨의 노드에서 세 가지 확장(decomposition)인 Shannon, positive Davio, negaive Davio중의 하나를 사용하도록 한다. 또한 이를 더욱 발전시킨 형태인 OPKFDD[3]는 각 노드마다 위의 세 확장 중에서 임의의 하나를 선택하도록 한 것이다. 따라서 DD는 각 노드의 확장 방법을 어떤 타입을 선택하느냐에 따라서 DD의 종류가 결정된다. 그러므로 주어진 불리안 함수를 최소의 노드를 갖는 DD로 표현할 때는 입력변수 순서와 각 노드의 확장형을 결정하는 방법을 고려해야 한다.

첫째, 입력변수 순서를 결정하는 방법은 다음과 같이 연구되어 왔다. OBDD에서의 입력 변수 순서를 결정하는 문제는 NP-complete 임이 증명되어 있으며[4] 따라서 이를 해결하기 위한 방법들은 여러 가지 휴리스틱(heuristic)에 의존하고 있다[5]. 또한 OFDD와 OKFDD의 최적의 입력변수 순서를 결정하는 연구 OFDD와 OKFDD의 최적의 입력변수 순서를 결정하는 연구[6]가 dynamic reordering [7]의 개념을 이용하여 이루어지고 있다. 이들 연구에서 제안한 방법은 sifting과 인접변수 교환 방법을 이용하며 OBDD와 비교해서 더 간소화된 함수표현이 가능하나 입력 변수가 많은 함수에 대해서는 실질적인 시간 내에 최적의 순서를 찾는다는 것은 매우 어려운 문제로 알려져 있다. 또한 이와 같은 문제를 해결하는 한 방법으로서 유전자 알고리즘을 적용한 연구[8]가 이루어지고 있다.

OBDD을 이용한 불리안 함수의 표현은 그림 1과 같이 함수 f가 $a_1b_1 + a_2b_2 + a_3b_3$라 할 때 (a)는 입력변수 순서가 a_1, b_1, a_2, b_2, a_3, b_3이고 (b)의 입력변수 순서는 a_1, a_2, a_3, b_1, b_2, b_3이다. 이때 함수 f를 주어진 입력변수 순서에 따라서 각 노드의 확장 방법을

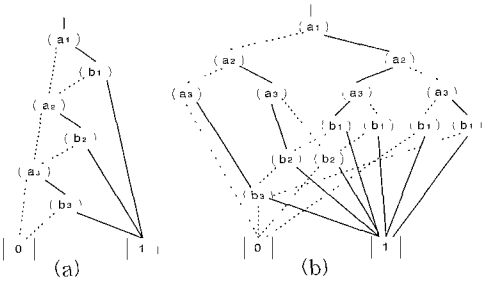


그림 1 입력변수 순서에 따른 OBDD의 크기

Shannon으로 확장한 경우 (b)의 입력순서에 비해 (a)의 입력변수 순서로 불리안 함수를 표현했을 때 OBDD의 노드 수가 작음을 알 수 있다. 따라서 DD을 이용한 불리안 함수의 최적화를 위한 입력변수 순서를 결정하는 방법은 중요한 문제가 된다. 마찬가지로 OPKFDD에서도 불리안 함수의 최적화 표현을 위해서 가장 적은 노드 수로 OPKFDD를 구성할 수 있는 입력변수 순서를 결정해야 한다.

둘째, 각 노드의 확장 방법은 세 가지 확장(decomposition)인 Shannon, positive Davio, 그리고 negaive Davio 확장 중 한가지를 선택한다. 각 노드의 선택의 폭이 넓어짐에 따라 표현공간 최소화의 가능성은 더욱 높아지나[9] OPKFDD의 경우 n 개의 입력변수에 대해서 3^{2^n-1} 개의 서로 다른 확장을 할 수 있으므로 이를 최소화하는 연구는 상대적으로 더 복잡성을 띄게 된다[3]. 예를 들어 만약 주어진 함수 f가 $ab' + ac' + a'bc$ 라 할 때 그림 2와 같이 같은 함수를 표현하는데 있어서 DD의 종류에 따라서 필요한 노드의 수가 다르다. (그림 2)의 (a)와 (b)를 비교해 보면 같은 함수를 각 노드의 확장 방법을 어떻게 선택하느냐에 따라 DD의 크기가 달라진다. (a)는 모든 노드를 shannon으로 확장하는 OBDD 형태로 표현한 경우이고 (b)는 각 노드마다 각기 다른 확장 방법을 사용하여 OPKFDD를 생성한 경우이다. OPKFDD에 의한 불리안 함수 표현은 각 노드마다 각기 다른 확장 방법을 선택할 수 있으므로 BDD보다 더 적은 노드 수로 함수를 표현할 수 있다. 따라서 같은 함수를 표현하지만 DD 종류에 따라서 즉, 각 노드의 확장형에 따라서 노드 수가 다름을 알 수 있다.

본 논문에서는 주어진 불리안 함수를 표현하기 위해서 OPKFDD를 사용한다. OPKFDD를 최소화하기 위해서는 위와 마찬가지로 두 가지 관점에서의 연구가 필요하다. 첫째, 입력 변수가 n개 일 때, $n!$가지의 입력 변수 순서 중에서 DD의 크기를 작게 하는 입력변수 순서

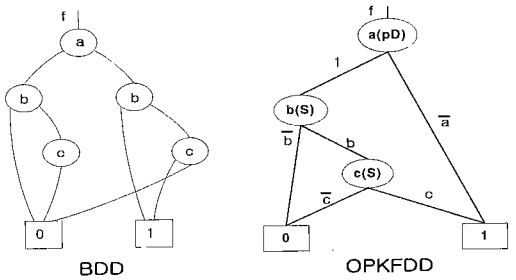


그림 2 각 노드의 확장방법에 따른 DD의 크기

를 결정하는 것이다. 둘째, OBDD, OFDD, 그리고 OKFDD와는 다르게 OPKFDD에 대한 연구는 각 노드에서의 확장방법을 선택하는 문제가 새로이 고려되어야 하며 이러한 특성 때문에 입력변수의 순서를 결정하는 것은 매우 어려운 문제로 알려져 있다[10]. 기존의 연구에서는 OBDD와 OKFDD에서 입력변수와 각 노드의 확장방법을 결정하기 위해서 유전자 알고리즘 방법이 적용되었으나 OPKFDD에서의 유전자 알고리즘 적용에 관한 연구가 아직 이루어지지 않고 있다. OPKFDD에서의 입력변수와 각 노드의 확장 방법을 병행해서 최적의 해를 구하기 위해서는 $n! \cdot 3^{2^n-1}$ 의 경우의 수를 고려해야 하며 입력변수 n에 따라서 해 공간이 지수적으로 늘어나게 된다. 이렇게 큰 해 공간에서 최적의 해를 탐색하기 위해서는 유전자 알고리즘 적용이 필요하다.

유전자 알고리즘(Genetic Algorithm)은 최적화 문제와 기계학습(machine learning)에서 자주 사용된다[11, 12]. 여러 가지 응용분야에서 전통적인 최적화 방법보다 우위에 있고 최근에는 CAD(Computer Aided Design)의 라우팅(routing), 배치(placement), 테스트 패턴 생성(test pattern generation) 그리고 논리합성(logic synthesis)과 같은 어려운 문제를 해결하는데 성공적으로 적용되고 있다[8]. 유전자 알고리즘은 자연의 유전학(natural genetics)과 자연선택(natural selection)의 원리에 바탕을 둔 통계적(stochastic) 최적해 탐색 방법을 이용한다. 기존의 최적화 방법에서는 해 공간이 너무 클 경우에는 탐색 영역이 방대해져 최적해 탐색이 불가능하다. 유전자 알고리즘은 기존의 최적해 탐색이 국부탐색이었는데 비하여 여러 해를 동시에 탐색하는 전역 탐색(global search)을 함으로써 전역적인 최적해(global optimal solution)를 찾을 확률이 기존의 최적화 탐색에 비해 큰 것이 특징이다. 이와 같은 원리를 입력변수 순서와 각 확장 방법을 결정하는 방법에 적용하고 있다. 이들 방법은 dynamic reordering 방법과 DTL- sifting [8]과 비교하여 많은 연산 시간이 소요되지만 보다 최적에 가까운 해를 구한다.

본 논문의 구성은 다음과 같다. 2장에서는 OPKFDD의 정의와 유전자 알고리즘의 개념을 제시하고, 3장에서는 유전자 알고리즘을 이용한 OPKFDD에서의 확장 방법과 입력변수 순서 결정 방법을 제안한다. 그리고 4장에서는 3장에서 제안한 알고리즘의 실험결과를 제시하며, 마지막으로 결론과 향후 연구 방향을 언급한다.

2. 관련연구

2.1 OPKFDDs(Ordered Pseudo-Kronecker Functional Decision Diagrams)

DD와 OPKFDD의 정의는 다음과 같다[5]. OPKFDD는 확장된 DD의 개념을 갖는다.

정의 1. Decision Diagram

입력변수 $X_n := \{x_1, x_2, \ldots, x_n\}$에 대한 DD는 루트가 있는 방향성 비 순환 그래프이다. 이 그래프는 2가지 타입의 노드를 갖는데 비-단말(non-terminal)과 단말(terminal) 노드로 구성된다. 비-단말 노드 v는 입력 변수 x_i로 분류되고, $low(v)$와 $high(v)$ 두 개의 자 노드를 갖는다. 그리고 단말 노드 v는 0 또는 1의 값을 갖고 자 노드가 없다.

정의 2. Ordered Decision diagram

DD의 각 경로에서 각 입력 변수가 최소한 한번 있고 각 경로에서 입력변수의 순서가 동일하다면 DD를 ordered하다고 한다.

DD는 아래와 같은 확장 타입을 사용하여 불리안 함수를 표현할 수 있다.

$$f = \overline{x_i} f_i^0 \oplus x_i f_i^1 \quad shannon\ (S) \tag{1}$$

$$f = f_i^0 \oplus x_i f_i^2 \quad positive\ Davio\ (pD) \tag{2}$$

$$f = f_i^1 \oplus \overline{x_i} f_i^2 \quad negative\ Davio\ (nD) \tag{3}$$

여기서 f_i^0와 f_i^1는 각각 $x_i=0$와 $x_i=1$에 대한 f의 *cofactor*이고, $f_i^2 = f_i^0 \oplus f_i^1$이다. 그리고 $\oplus$은 Exclusive OR연산자이다.

정의 3. OPKFDD

OPKFDD는 입력변수 x_n에 대한 ordered DD이고 각 노드마다 Shannon, positive Davio, 그리고 negative Davio 확장 중에서 한 가지를 선택하여 각기 다른 확장 타입으로 구성된다.

예를 들면 입력변수가 n개인 경우 함수를 그림 3과 같이 OPKFDD로 구성할 수 있다. OPKFDD에서 각 노드는 서로 다른 확장 방법을 선택할 수 있으므로

OPKFDD는 3^{2^n-1}개의 서로 다른 확장 방법이 있을 수 있다.

정의 4. OPKFDD의 노드

OPKFDD에서 어떤 노드가 Shannon expansion으로 확장되면 Shannon 노드라고 하고 Davio expansion 중에서 positive Davio expansion으로 확장된다면 그 노드를 positive Davio 노드라고 하며, nagative Davio expansion으로 확장된다면 그 노드를 negative Davio 노드라고 한다.

정의 5. DD 크기

어떤 불리안 함수 F의 DD 크기를 $|F|$라 할 때 이 크기는 F의 단말노드를 제외한 노드 수이다.

OPKFDD의 표현은 보수간선(Compl-emented Edges)을 사용함으로써 그 크기를 감소시킬 수 있다. 만약 함수 f의 OPKFDD가 부 그래프 g와 g'를 갖는다면, g와 g'의 보수관계를 이용하여 DD에서 두 개의 부 그래프를 유지하지 않고 보수간선을 이용하여 하나의 부 그래프만을 유지함으로써 DD의 전체 크기를 줄일 수 있다[3].

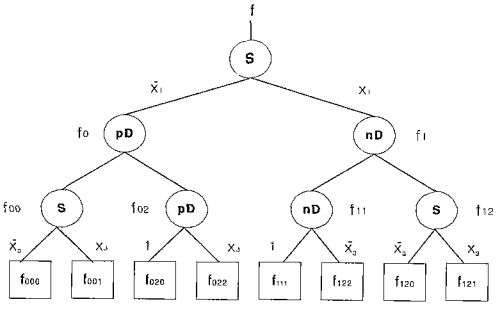


그림 3 n변수 함수에 대한 OPKFDD의 확장방법 선택

2.2 유전자 알고리즘의 개념[12]

유전알고리즘은 1970년대 미국의 John Holland 교수에 의해 정립된 이론으로 자연의 유전학(natural genetics)과 자연선택(natural selection)의 원리에 바탕을 둔 통계적(stochastic) 최적 해 탐색 방법이다. 해 공간이 너무 클 경우에는 탐색 영역이 방대해져 최적 해 탐색이 불가능하다. 기존의 최적 해 탐색이 국부탐색이었는데 비하여 유전알고리즘은 여러 해를 통시에 탐색하는 전역 탐색(global search)을 함으로써 전역적인 최적 해(global optimal solution)를 찾을 확률이 기존의 최적화 탐색에 비해 큰 것이 특징이다.

유전자 알고리즘은 해결해야 할 문제공간을 GA공간으로 기호화(encoding)하여 GA공간상에서 유전자 연산자를 적용하면서 최적의 해를 구한다. GA공간상에서 개체의 전이는 교차 및 돌연변이에 의해서 규정된다. 하나의 개체가 한번의 돌연변이에 의해서 전이하여 얻는 부분 공간을 돌연변이 근방(mutation neighborhood)라 부르고, 두 개의 개체로부터 결정된 교차방법 하에서 한 번의 교차에 의해서 전이하여 얻는 부분공간을 교차 근방(crossover neighborhood)이다.

GA공간상에서 개체 P_1과 P_2사이의 교차에 의해서 개체 C가 생성되고, 이것을 역 사상함으로써 문제공간상에 부모 P_1과 P_2의 형질을 부분적으로 계승한 개체 C가 생성되는 것을 그림 4에서 보여주고 있다. 여기서 고려해야할 점은 코드화 및 교차가 부적절하다면 GA공간에서 교차에 의해 생성된 개체를 문제공간으로 역사상할 때, 실행 불가능한 해가 될지도 모른다. 여기에서 실행 불가능한 해를 만들어 내는 염색체는 치사 유전자를 갖는다고 한다. 또한 실행 가능하다고는 해도 양쪽 부모와는 완전히 성격이 다른 개체가 생성되어버릴 수도 있다. 따라서 입력변수에 대한 교차 연산을 할 경우에 교배연산 후 새로 생성된 염색체가 문제공간에서 적절한가를 검사하고 만약 오류가 발생하면 그것을 회복시킬 복구 알고리즘이 필요하다.

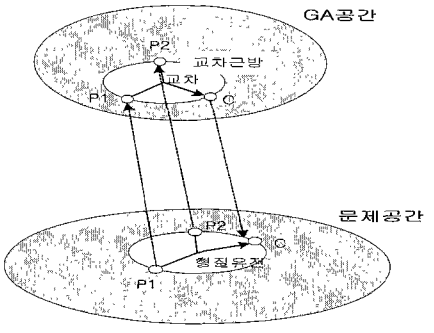


그림 4 문제공간과 GA공간의 대응관계

유전자 알고리즘에 의한 탐색 또는 최적화 문제의 해결에 있어서 전역탐색(expolration)과 지역탐색(exploitation)의 조화가 중요하다. 획득한 정보의 유효한 이용은 기존의 등고선법(hill-climbing)과 유사해 지며, 미지의 영역에 대한 탐색이 강조 될수록 랜덤탐색(random search)과 같은 특성을 나타내게 된다. 유전자 알고리즘은 이러한 두 가지의 조건을 함께 제어할 수 있는 매력적인 알고리즘이다. 이를 제어하기 위한 파라미터에는 여러 가지가 있으나 중요한 것 3가지만 들자면 개체군

의 크기(population size : M), 교배 확률(probability of crossover: P_c), 돌연변이 확률(probability of mutation: P_m) 이다.

3. 유전자 알고리즘을 이용한 OPKFDD의 최소화

OPKFDD에서는 OKFDD와는 다르게 각 노드에서의 확장방법을 선택하는 문제가 새로이 고려되어야 하며 이러한 특성 때문에 입력변수의 순서를 결정하는 것은 매우 어려운 문제이다. 즉, OPKFDD의 크기는 입력변수 순서와 각 노드의 확장방법을 어떻게 선택하는 가에 따라 달라진다. 입력변수 순서와 각 노드의 확장 방법을 병행하여 고려할 때는 다음과 같이 다양한 크기를 갖게 된다. 첫째, 주어진 확장타입에 대해서 OPKFDD의 크기는 입력변수의 순서를 변화시켜 줌으로써 선형(linear)부터 지수(exponential)적 증가율을 가지고 매우 다양하게 변화한다. 둘째, 주어진 입력변수 순서에 대해서 OPKFDD의 크기는 확장 타입을 어떻게 선택하느냐에 따라 다항(polynomial)부터 지수적 증가율을 가지고 다양한 크기를 가진다. 이렇게 다양하고 큰 해 공간에서 최적의 해를 탐색하기 위해서는 유전자 알고리즘 적용이 필요하다. 본 논문에서는 OPKFDD를 최소화하기 위해서 입력변수 순서와 각 노드의 확장 방법을 병행하여 결정하고 보다 빠르게 최적의 해에 수렴하기 위하여 유전자 알고리즘의 처음에 [13]에서 제안한 방법을 적용하는 혼합(hybrid) 유전자 알고리즘을 제시한다. [13]에서는 최소의 노드 수를 갖는 OPKFDD를 구성하기 위해서 입력변수 순서와 노드의 확장 방법을 결정하는 방법을 제안했다. 첫째, 입력변수 순서 결정은 함수간의 포함관계를 이용한 입력변수 순서 결정 방법을 제안하고 있다. 이 방법에서는 다 출력 함수의 입력변수 순서 결정은 함수들의 각각을 가지고 결정하기보다는 함수간의 포함 관계를 고려하여 결정한다. 함수간의 포함관계는 support를 이용하여 구하며 이를 하나의 그룹으로 간주하여 그룹간의 부분적인(partial) 순서를 결정하고 마지막으로 각 그룹내의 입력변수들을 sifting 알고리즘을 이용하여 수정하여 나감으로써 최종적인 입력 변수의 순서를 결정한다. 두 번째, 각 노드의 확장 방법은 주어진 OBDD의 각 노드에 대한 cofactor $f_{v=1}$와 $f_{v=0}$을 exclusive-XOR($f_{v=0} \oplus f_{v=1}$)하여 THEN-간선과 ELSE- 간선 외에 추가의 XOR-간선을 만든다. 이렇게 수정된 OBDD에서 OPKFDD에서 사용하는 *S*, *nD*, 그리고 *pD*의 세 가지 확장방법은 이 세 간선을 이용하여 구현할 수 있으므로 이 세 간선 중에서 두 개를 선택한다. 이를 위하여 각 노드마다 비용함수를 설정하여 최소의 complex term을 생성하도록 각 노드의 확장방법을 결정한다. [13]에서 제안한 방법에서는 해 공간이 큰 이유로 먼저 최적의 입력변수 순서를 구하고 구해진 입력변수 순서에서 각 노드의 확장 방법을 결정한다.

3.1 개체 표현

3.1.1 입력변수에 대한 개체 생성 방법

입력변수를 위한 초기 개체(Initialization Population)는 입력변수의 순서를 나타내는 π-배열(array)로 나타낸다. 이 배열은 정수 문자열(integer string('1234'))로써 표현되고 오름차순 형태를 갖는다. 또한 π-배열의 크기는 입력변수 크기로 한다.

초기 개체는 입력변수 2배수를 생성하고 랜덤하게 생성한다. 생성 방법은 각 배열의 요소에 랜덤 정수가 할당하여 저장하고 랜덤 정수 순서로 정렬하여 만든다. 그림 5는 입력변수 수가 6개인 함수에 대한 각 배열 요소에 난수를 발생시키는 것과 그 난수를 정렬하여 초기 개체 생성에 대한 예제이다.

배열첨자	1	2	3	4	5	6
난 수	76	214	14	309	290	121

배열첨자	3	1	6	2	5	4
난 수	14	76	121	214	290	309

그림 5 입력변수에 대한 기호화

3.1.2 확장형 결정에 대한 개체생성 방법

각 노드의 확장 방법을 선택하기 위한 개체는 확장방법의 타입을 나타내는 d-배열로 나타낸다. 그림 6과 같이 배열은 각 노드의 확장 타입인 S, pD 그리고 nD의 3가지의 값을 갖는 정수로 표현한다. 그리고 배열의 크기는 노드 수로 한다. 초기 개체는 π-배열과 마찬가지로 입력변수의 2배수를 생성하고 랜덤하게 생성한다.

nD	pD	S	S
2	1	0	0

그림 6 노드의 확장 방법에 대한 기호화

3.1.3 유전 연산자

1) 입력변수에 대한 유전 연산자

① 재생산

본 논문에서는 새로운 개체를 생성하기 위해서 GA공

간의 개체들 중에서 유전 연산을 하기 위한 부모를 선택하기 위해서 Roulette Wheel Selection 방법을 사용하여 적합도가 높은 개체의 유전 인자가 다음세대 개체에 영향을 줄 수 있도록 한다. 그리고 현재세대의 개체들 중에서 다음세대에 계승할 개체들의 결정은 Steady-State Selection 방법을 사용하여 현재세대의 개체 중에서 적합도에 따라서 상위 20%의 개체를 복사하여 다음세대에 계승한다.

② 교배 연산자

교배 연산 단계에서 개체집단을 모두 연산하는 것이 아니라 개체의 1/2만 적용한다. 연산과정은 그림 7과 같이 GA공간에서 임의의 두 개의 개체 P_1과 P_2를 선택하여 유전자 알고리즘의 교배 연산자 중 부분사상교배연산자(PMX:Partially Matched Crossover)에 적용한다. PMX는 두 개의 부모로부터 두 개의 자식을 생성하는데 두 부모의 절단(cut) 위치사이의 부분을 서로 교환하여 두 개의 자식 C'_1와 C'_2가 생성한다. C'_1 염색체는 입력변수 순서 5와 4가 중복이 되고 C'_2염색체는 입력변수 순서 2와 6이 중복되는 치사 유전자를 갖게 된다. 이것은 새롭게 생성된 C'_1과 C'_2를 문제공간에 역사상했을 때 실행 불가능한 해가 된다. 이 경우에 올바른 해가 되도록 하는 별도의 과정, 즉 복구 알고리즘을 이용하여 거쳐 정상화한다. 그래서 최종적인 두 개의 자식 C_1과 C_2를 생성한다.

P_1	2 0 3	4 5	1 6
P_2	5 3 1	2 6	0 4
C'_1	**5 3 1**	**4 5**	0 **4**
C'_2	**2 0 3**	**2 6**	1 **6**
C_1	6 3 1	4 5	0 2
C_2	4 0 3	2 6	1 5

그림 7 부분사상교배연산자(PMX)의 연산과정

③ 돌연변이 연산자

돌연변이 연산자를 통해서 집단 중에서 유전자형의 다양성을 확보하고자 10%의 돌연변이 율을 가지고 수행한다. 돌연변이 연산에 대한 예제는 다음과 같다. 임의의 부모를 하나 선택하고 한 위치를 선택한다. 그리고 새로운 값을 랜덤하게 결정(valid range에서)하고 그 값을 포함하는 위치를 선택한다. 그리고 나서 두 개의 위치에 값을 서로 교환한다. 연산과정은 그림 8과 같이 GA공간에서 한 개의 개체 P를 선택하고 입력변수 중에서 랜덤하게 교체할 6과 2를 선택하고 염색체 P의 입력변수 순서에서 그에 해당되는 각각의 위치를 구한 다음에 두 개의 위치에 대응되는 입력변수를 서로 교환하여 돌연변이를 발생시킨다.

P	6	3	1	2	5	0	4
C	2	3	1	6	5	0	4

그림 8 돌연변이 과정

2) 각 노드의 확장방법 결정을 위한 유전 연산자

OPKFDD는 다른 DD와는 다르게 각 노드마다 다른 확장 방법을 사용하여 불리안 함수를 표현할 수 있고 확장 방법 선택에 따라서 그 크기가 달라짐을 알 수 있다. 따라서 입력 변수 순서에 따른 노드의 확장 방법을 결정해야 한다. 각 노드의 확장 방법을 결정하기 위한 유전 연산은 입력변수 순서 결정을 위한 유전 연산을 적용하고 난 후 실행한다. 확장 방법에 대한 문제공간의 해를 기호화한 GA공간인 d-배열에 대한 유전 연산자와 예제는 다음과 같다.

① 재생산

본 논문에서는 새로운 개체를 생성하기 위해서 GA공간의 개체들 중에서 유전 연산을 하기 위한 부모를 선택하기 위해서 Roulette Wheel Selection 방법을 사용하여 최소의 노드 수를 갖는 각 노드의 확장 방법과 현재세대의 개체들 중에서 다음세대에 계승할 개체들의 결정은 Steady-State Selection 방법을 사용하여 현재세대의 개체 중에서 적합도에 따라서 상위 20%의 개체를 복사하여 다음세대에 계승한다.

② 교배 연산자

GA공간상의 d-배열에서 임의의 두 개의 개체를 선택하여 교배연산을 위해 다음과 같이 수행된다. 교배 연산자는 GA공간상의 개체들 중에서 적합도가 높은 개체들이 유전 연산을 할 수 있는 확률을 높여서 그 중에서 랜덤하게 두 개의 개체를 부모로 선택한다. 그리고 교배위치를 결정하여 교배위치에서 두 부모개체들의 유전자를 서로 교환한다. 예를 들면 그림 9와 같이 부모 개체 P_1과 P_2이 있을 때 P_1과 P_2의 절단 위치에서 P_1의 두 번째 부분과 P_2의 두 번째 부분을 서로 교환하여 다음 자식 새대 C_1과 C_2을 생성한다.

P_1	0 **0 2**
P_2	1 **1 0**
C_1	0 1 0
C_2	1 0 2

그림 9 교배 연산 과정

③ 돌연변이 연산자

본 논문에서는 각 노드의 확장 방법을 결정하기 위해

서 10%의 돌연변이 율을 가지고 수행한다. 돌연변이 연산은 다음과 같이 수행된다. 랜덤하게 돌연변이 연산을 수행 할 개체와 돌연변이 위치를 선택하여 값을 변화시키고 나머지는 그대로 복사하여 새로운 요소를 생성한다. 예를 들면 그림 10과 같이 GA공간에서 선택된 부모개체 P의 첫 번째 부분을 돌연 변이 위치로 결정하고 첫 번째 확장 방법을 shannon에서 negative Davio expansion으로 바꾼다. 즉 기호 값 0에서 2로 하여 다음세대 C을 생성한다.

P	0 0 2
C	2 0 2

그림 10 돌연변이 연산 과정

3.1.4 알고리즘

문제공간의 입력변수 순서와 각 노드의 확장 방법을 GA공간상의 개체로 나타내기 위해서 기호화하여 초기 개체를 생성하고 위의 유전자 연산자를 이용하여 그림 11과 같이 유전자 알고리즘을 수행한다. 그 결과로 최소의 노드 수를 갖는 OPKFDD를 생성하기 위한 입력변수 순서와 그에 따른 각 노드의 확장 방법을 결정한다.

```
genetic_algorithm(benchmark)
{
  /* 초기 개체생성 */
  generate_random_population();
  calculate_fitness(); /* 적합도 계산 */
  /* 초기 개체에 GroupSift-DTL 알고리즘 적용 */
  optimize_initial_population_with_GroupSift-DTL();
  {
    do
    {
      /* 유전 연산자 적용 */
      apply_operators_with_corresponding_probabilities();
      /* 새로 생성된 요소에 대한 적합도 계산 */
      calculate_fitness();
      /* 자식 세대 생성 */
      update_population(child);
    } while(not terminal case);
     /* 지정된 반복회수만큼 수행 */
   return;
}
```

그림 11 유전자 알고리즘

위의 알고리즘은 다음의 단계로 수행한다. 1 단계에서는 입력변수 개수의 2배로 GA공간상의 두 종류의 개체군을 생성한다. 그 개체는 입력변수 순서를 나타내는 π-배열과 입력변수 순서에 따른 각 노드의 확장 방법을 나타내는 d-배열의 초기 개체군을 생성한다. 2 단계에서는 해(solution)에 빠른 수렴을 위하여 초기 개체에 [13]에서 제안한 GroupSift-DTL 알고리즘을 적용한다. 3단계에서는 각 요소에 입력변수 연산자(Reproduction, PMX)를 적용한다. 그리고 나서 4단계로 입력변수 연산자중 돌연변이 연산자를 적용한다. 입력변수 순서 결정을 위한 연산 후에 5단계에서는 d-배열에서 선택된 개체에 DTL 연산자(재생산, 교배)를 적용하여 새로운 요소를 생성하고 새롭게 생성된 요소는 주어진 확률에 따라서 각 노드의 확장 방법 연산자(돌연변이 연산자)를 적용하여 수정된다. 유전 연산이 끝나면 6단계에서 적합도에 따라서 다음세대에 유전될 개체를 선택한다. 새로운 세대가 생성되면 다시 3단계부터 다시 반복한다.

3.1.5 적합도 함수

d-배열과 π-배열에 위의 유전자 연산자를 이용하여 유전자 알고리즘을 적용하여 다음 세대의 개체를 생성하는 기준이 되는 것이 적합도 함수이다. 본 논문에서의 적합도 함수를 노드 수로 한다.

3.1.6 입력 파라미터 설정

유전자 알고리즘은 초기 개체와 유전 연산자를 잘 표현하는 것도 중요하지만 무엇보다도 입력 파라미터를 어떻게 결정하느냐에 따라서 수행 결과에 많은 영향을 준다. 본 논문에서 사용한 파라미터 종류와 파라미터의 디폴트값 설정은 표 1과 같다. 실험결과에서 각 파라미터의 값의 변화에 따른 결과를 비교 분석한다.

표 1 파라미터 종류와 값

파라미터 종류	파라미터 값
개체 크기	입력변수의 2배
재생산율	20%
교배율	90%
돌연변이율	10%

4. 구현 및 실험결과

본 논문에서 제시한 방법은 C언어로 구현했고, SUN UltraSpac Ultima1에서 실행했다. 실험 결과는 주어진 벤치마크(benchmark) 회로에 대하여 유전자 알고리즘을 적용한 결과를 제시한다. 그리고 실행 시간은 CPU time으로 하고 단위는 초(second)이다.

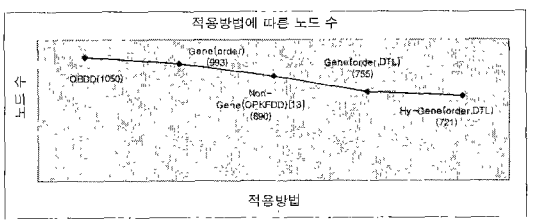
표 2는 유전자 알고리즘을 적용한 결과를 나타내고 표 3은 그에 따른 통합표를 나타낸다. 세 번째 열은 OBDD와 같이 각 노드의 확장 방법을 shannon으로 고정하고 유전자 알고리즘에 입력변수만을 고려한 결과이고 네 번째 열은 유전자 알고리즘에 입력변수 순서와 각 노드의 확장방법을 병행으로 결정하는 실행 결과를

나타낸다. 그리고 첫 번째 열과 세 번째 열은 각각 OBDD와 OPKFDD를 생성하는데 유전자 알고리즘을 적용하지 않은 결과이다. 여기에서 OPKFDD는 [13]의 방법을 이용해서 OPKFDD를 생성한 결과를 나타낸다. 위의 결과에서 알 수 있듯이 유전자 알고리즘을 적용한 경우가 OBDD와 비교해서 30% 그리고 OPKFDD와 비교해서 18% 노드 수가 적어졌음을 알 수 있다. 또한 초기에 생성한 개체를 바로 유전자 알고리즘을 적용한 것보다 초기의 개체에 [13]에서 제안한 방법을 먼저 적용하고 그 후에 유전자 알고리즘을 실행한 결과가 더 나은 결과를 보이고 해에 수렴 속도가 개선되었다.

표 2 유전자 알고리즘을 적용한 방법

Bench-mark 회로	Non-Gene (OBDD)		Gene (order)		Non-Gene [13] (OPKFDD)		Hy-Gene (order ,DTL)		Gene (order, DTL)	
	#node	time	#node	time	#node	time	#node	time	#node	time
5xp1	32	1.0	32	1.6	32	1.0	20	8.5	28	8.7
card4	23	1.0	22	3.2	19	0.9	15	10.8	23	17.1
clip	152	6.9	113	7.9	132	4.3	73	43.3	79	87.4
clog8	192	6.3	159	8.9	165	4.9	155	147.0	160	166.9
cmlp4	146	4.0	145	6.7	101	3.2	113	103.4	113	111.7
cnrm	118	2.0	118	8.1	123	2.0	133	138.5	143	150.3
cu	34	1.0	29	4.9	34	1.0	17	25.4	20	27.5
f51m	75	0.37	34	1.9	45	0.33	20	10.1	23	12.9
inc	81	1.7	69	2.3	75	1.6	57	26.8	62	28.5
mlp3	39	1.0	35	1.2	27	1.0	23	6.7	26	7.1
rd53	14	0.9	14	1.1	12	0.9	11	4.4	11	4.3
rd73	28	0.9	28	1.6	24	0.9	20	20.6	20	20.8
sao2	86	1.6	30	7.3	85	1.6	49	28.5	51	41.6
t481	30	1.3	105	1.3	16	1.0	15	142.9	16	142.9
total	1050	29.9	933	58.0	890	24.63	721	716.9	775	827.7

표 3 적용방법에 따른 통합표



다음은 유전자 알고리즘의 수행 결과에 영향을 주는 중요한 인자인 파라미터 값에 따른 실험결과를 나타낸다. 표 4는 개체 수의 값 변화에 따른 결과이다. 이 실험 결과는 교배 율과 돌연변이 율은 각각 90%와 10%로 고정시키고 개체 수에 변화시켜 수행시킨 결과이다. 이 결과에서 각 세대별 개체 수를 크게 하는 경우가 그렇지 않는 경우보다 노드 수가 적어짐을 알 수 있다. 이것은 전체 해 집단에서 실험에서 대상 개체가 많다는 것은 전역적 탐색에 가까워진다는 것을 알 수 있고 대신에 수행 속도 면에서 고려해보면 개체 수가 많아질수록 연산속도가 늦어진다. 따라서 개체 수는 유전자 알고리즘의 수행 결과에 영향을 줌을 알 수 있다.

표 4 개체 수 변화에 따른 유전자 알고리즘의 수행 결과

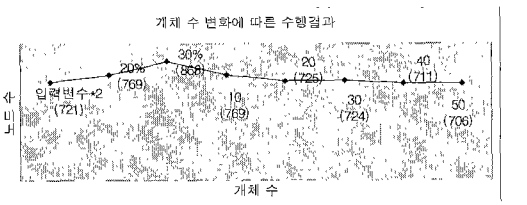


표 5 교배 율 변화에 따른 수행 결과

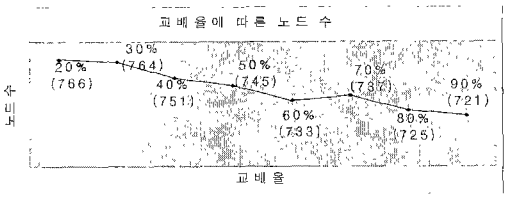


표 6 돌연변이 율에 따른 실험결과

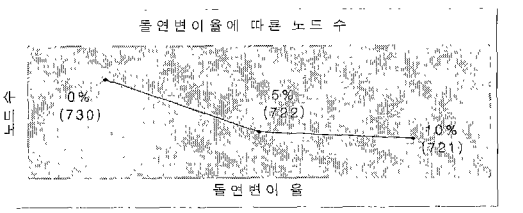


표 5는 교배 율에 따른 실험결과이다. 이 실험 결과는 개체 수와 돌연 변이 율을 각각 입력변수의 2배와 10%을 고정시키고 교배 율의 변화에 따른 수행 결과이다. 이 결과를 보면 이 교배 율을 높게 함으로써 20%의 교배 율과 비교해서 90%일 때 6%의 개선된 결과를 보인다.

표 6은 돌연변이 율에 따른 실험결과이다. 이 실험결과는 개체 수와 교배 율을 각각 입력변수 수의 2배와 90%로 고정시키고 여러 가지 교배 율에 따른 실험결과이다. 이 실험에서 보듯이 돌연변이 연산을 하지 않는 경우보다 돌연변이 연산을 한 실험결과가 개선된 결과을 보인다.

5. 결론 및 연구방향

본 논문에서는 OPKFDD을 이용한 불리안 함수의 최적화 표현을 위해서 입력변수 순서와 각 노드의 확장 타입 선택을 병행으로 실행할 수 있도록 유전자 알고리즘을 제안하였다.

이 방법에서는 문제공간의 입력변수 순서와 각 노드의 확장 방법을 각각 GA공간의 π-배열과 d-배열로 해 집단의 개체를 만들었다. 생성된 개체를 교배 연산자와 돌연변이 연산자를 이용하여 다음세대의 개체를 생성하였다. 교배연산자를 통해서 한 세대의 좋은 결과에 대한 입력변수 순서와 확장방법에 대한 정보를 다음 세대에 유전시키고 전역 최적화를 가능하게 하고 지역 최적화를 방지하기 위해서 돌연변이 연산을 수행하였다. 또한 초기 개체 생성을 위해 [13]을 유전자 알고리즘에 적용하여 혼용 유전자 알고리즘을 제안하였다.

이 실험에서는 유전자 알고리즘의 수행 결과에 영향을 주는 파리미터, 즉 개체 수, 교배 율 그리고 돌연변이 율의 값에 따른 실험을 해보았고 첫 번째 방법을 유전자 알고리즘의 처음에 수행함으로써 처음 생성되는 개체가 해에 수렴을 빨리 할 수 있는 좋은 개체군부터 수행을 할 수 있도록 했다. 유전자 알고리즘을 적용한 방법은 첫 번째 방법과 비교해 보면 노드 수에서 더 개선된 결과를 보이지만 수행속도 면에서 보면 유전자 알고리즘 특성상 많은 연산 시간을 요구했었다.

향후 연구 방향으로는 불리안 함수의 최적화 표현을 위해서 노드 수를 감소시킬 수 있도록 입력 변수의 순서를 찾는 알고리즘에 대한 연구와 본 논문에서 제안한 방법들이 최적의 해에 얼마나 근사한지 평가, 보완하는 연구가 이루어져야한다.

참 고 문 헌

[1] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," IEEE *Trans. on Computer*, pp.677-691, 1986.

[2] B. Becker, R. Drechsler, and M. Theobald. "On the Implementation of a Package for Efficient Representation and Manipulation of Functional Decision Diagrams," *IFIP Workshop on the Applications on the Reed Muller Expansions*, pp.162-169, 1993.

[3] T. Sasao and M. Fujita, Representations of Discrete Functions, *Kluwer Academic Publishers*, 1996.

[4] B. Bollig, P. Savicky, and I. Wegener, "On the Improvement of Variable Orderings for OBDDs," *IFIP Workshop on Logic and Architecture Synthesis, Grenoble*, pp.71-80, 1994.

[5] S. Malik, A. Wang, R. Brayton, and A. S. Vincentelli, "Logic Verification Using Binary Decision Diagrams in Logic Synthesis Environment," *In Proceedings International Conference on Computer-Aided Design*, pp.6-9, Nov. 1988.

[6] B. Becker, R. Drechsler, and M. Theobald. "On the Implementation of a Package for Efficient Representation and Manipulation of Functional Decision Diagrams," *IFIP Workshop on the Applications on the Reed Muller Expansions*, pp.162-169, 1993.

[7] R. Rudell, "Dynamic Variable Ordering for Ordered Binary Decision Diagrams," *In IEEE International Conference on Computer-Aided Design*, pp.42-47, 1993.

[8] R. Drechsler, B. Becker and N. Drechsler, "Minimization of OKFDDs by Genetic Algorithms," *International Symposium on Soft Computing, Reading*, Vol. B pp.528-263, 1996.

[9] R. Drechsler, and B. Becker, "On Variable Ordering and Decomposition Type Choice in OKFDDs," *IEEE Tran. on Computer*, pp.1398-1403, Nov. 1998.

[10] R. Drechsler, personal communication, 1999.

[11] B. Bhanu and S. L. Lee, Genetic Learning for Adaption Image Sequentation, *Kluwer Academic Publishers*, 1994.

[12] D. E. Goldberg, Genetic Algorithms, *Addison Wesley*, 1989.

[13] 정미경, 이혁, 이귀상, "OPKFDD의 최적화 표현을 위한 decomposition과 입력변수 순서 결정", *한국정보처리학회 추계학술발표논문집*, 5권2호, pp. 1998.

정 미 경

정보과학회논문지 : 시스템 및 이론
제 27 권 제 1 호 참조

신 윤 정

1994년 목포대학교 컴퓨터공학과 졸업(학사). 1998년 경희대학교 교육대학원 전산교육전공(석사). 1999년 ~ 전남대학교 전산통계학과 박사과정. 1999년 ~ 현재 광주대학교 컴퓨터전자통신공학부 전임강사. 관심분야는 멀티미디어 영상처리, VLSI CAD

이 귀 상

정보과학회논문지 : 시스템 및 이론
제 27 권 제 1 호 참조

장 준 영

1985년 전남대학교 전산통계학과 졸업(학사). 1987년 중앙대학교 대학원 전산계산학과 석사. 1996년 중앙대학교 대학원 전산통계학과 박사. 1998년 전자통신연구원 회로소자연구소 POST-DOC. 1999년 ~ 현재 전자통신연구원 집적회로연구부 선임연구원. 관심분야는 CAD/VLSI, 논리합성, Codesign, Embedded system 설계, SOC 설계등임

배 영 한

1985년 한양대학교 전자공학과 졸업(학사). 1987년 한양대학교 대학원 전자공학과 석사. 1987년 ~ 현재 전자통신연구원 집적회로설계연구부 선임연구원, 시스템설계잔동화팀 팀장. 관심분야는 VLSI/CAD, 논리합성, HW/SW Codesign, Embedded system 설계, Embedded Processor 설계

조 한 진

1982년 2월 한양대학교 전자공학과 졸업. 1987년 미국 New Jersey Institute of Technology 전자공학과 석사. 1992년 미국 University of Florida 전자공학과 박사. 1992년 11월 ~ 현재 한국전자통신연구원 근무, 집적회로설계연구부 부장. 주관심분야는 VLSI design, 시스템 설계, TCAD 분야 등임