論文95-32B-1-2

유향 그래프의 최대 경로 길이를 제한하는 최소 노드 집합을 구하는 알고리즘

(Determining Minimal Set of Vertices Limiting The Maximum Path Length in General Directed Graphs)

李 東 浩 *
(Lee Dong Ho)

요 약

사이클이 있는 일반 그래프에서 최소한의 노드들을 제거하여 최대 경로의 길이를 주어진 값 이하로 제한하는 최소의 노드 집합을 구하는 문제를 정의하였다. 이 문제를 위한 최적 알고리즘(optimal algorithm)과 근사 알고리즘(heuristic algorithm)을 제안하고 대형 논리회로에서 추출한 그래프를 이용한 실험으로 근사 알고리즘이 매우 효과적임을 보였다. 근사 알고리즘은 그래프 리덕션(graph reduction)을 이용하여 최소 귀환 노드 집합(feedback vertex set)을 구하는 방법에 기초를 두고 있다.

Abstract

A new graph problem is formulated to limit the maximum path length of a general directed graph when a minimal set of vertices together with their incident edges are removed from the graph. An optimal algorithm and a heuristic algorithm are proposed and the proposed heuristic algorithm is shown to be effective through experiments using a collection of graphs obtained from large sequential circuits. The heuristic algorithm is based on a feedback vertex set algorithm based on graph reduction.

I. 서 론

본 논문은 사이클이 있는 일반 방향성 그래프에서 최소한의 노드들을 제거하여 사이클이 없고 최대 경로의 길이(maximum path length)가 주어진 값 이하로 제한되는 최소의 노드 집합을 구하는 문제에 관한 것이다. 이 문제는 비교적 생소한 문제로 순차적 논리회로에서 최선의 스캔 기억 소자를 선택하는데 효과적인 방법을 연구하는 과정에서 발견되었다[1]. 관련된 문제인 최소 귀환 노드 집합 문제(feedback vertex set problem)는 매우 중요한 문제로 공학에서 널리 사용된다[2]. 또한 제안된 근사적 방법은 최소 귀환 노드 집합을 구하는데 사용된 방법의 일종인 그래프 리덕션(graph reduction) 방법을 주 계산 방법으로 사용한다. 따라서 최소 귀환 노드 집합 문제에 대한 문

* 正會員, 慶北大學校 電子工學科
(Dept. of Elec. Eng., Kyungpook Nat'l Univ.)
接受日字 : 1994年 5月 28日

헌을 조사하는 것이 순서이다.

최소 귀환 노드 집합 문제는 매우 중요한 그래프 문제이다. 이 문제는 NP-complete 문제로 알려져 있고[3], 모든 최소한의 사이클(minimal cycle)을 생성하고, 그들을 자름에 의하여 근사적으로 구하여져 왔다[4]. 최소 사이클의 자세한 정의는 다음 절에서 주어진다. 최근에는 많은 저자들이 다항식 계산 시간(polynomial computation time)에 그러한 집합을 발견하는 것이 가능한 그래프의 클래스 발견하는데 성공하였다. 첫번째 결과의 하나는 Shamir에 의한 것이다[5]. Shamir는 리덕션 가능한 흐름 그래프(reducible flow graphs)로 알려진 그래프의 클래스에서 이 문제를 푸는 선형 알고리즘(linear time algorithm)을 발견하였다. Wang 등은 사이클릭 리덕션 그래프(cyclically reducible graphs)를 정의하고 간단한 다항식 시간 알고리즘을 발견하였다[6]. Lloyd 등은 그의 최근의 논문에서 더 많은 클래스를 발견하였다[2]. 또한 Rosen은 Shamir의 알고리즘이 더 일반적인 클래스의 그래프에 적용될 수 있으며 임의의 그래프에 적용되어도 최선의 결과를 얻을 수 있음을 보였다[7]. 특히 그는 간단한 후처리 방법을 사용하여 결과를 향상 시킬 수 있음을 보였다.

이와는 반대로 Smith 등은 일반 유향 그래프를 위한 최적 알고리즘(optimal algorithm)에 관한 연구를 하였다[8]. 현재 그의 알고리즘이 일반 그래프의 경우 최선의 알고리즘으로 알려져 있다. 그러나 그의 알고리즘은 최악의 경우 지수적 계산 시간(exponential computation time)을 가지고 있다. 마지막으로 Levy등은 그래프 축약(graph contraction)에 근거한 알고리즘을 보였다[9]. 그의 알고리즘은 본 논문에서 사용되는 그래프 리덕션 방법과 유사하다. 다만 그들은 본 논문에서 적용하는 선처리, 후처리 방법을 적용하지 않았을 뿐만 아니라 그들의 알고리즘의 효율성을 보일 수 있는 실험 자료를 제공하지 않았다.

본 논문에서는 순차적 논리회로(sequential logic circuit)의 부분 스캔 설계를 위하여 회로를 그래프 이론적으로 추상화한 S-그래프(S-graph)를 연구 자료로 사용하였다[10]. 다음에 본 논문에서 사용할 정의들을 기술한다.

정의 1. S-그래프는 순차적 논리회로와 관련된 유향 그래프(directed graph)이다. 노드 vi는 기억 요소를 나타내며 어떤 기억요소(memory element) i로부터 기억 요소 j까지 조합회로(combinational circuit)를 통한 경로가 있으면 에지(edge)가 존재한다[10].

정의 2. 어떤 노드 v1으로부터 vn까지의 경로, P(v1,vn)은 노드의 순서열 (v1, v2, ..., vn)으로 각 노드 vi로부터 vi+1로 에지가 있다. 여기서 i는 1부터 n-1이며 경로의 길이는 순서열에 포함된 노드 마이너스 1 이다.

정의 3. S-그래프 G의 최대 경로 길이, dmax(G)는 G 안에서의 모든 노드의 쌍들 간의 최대 경로 길이이다. 그래프 G가 빈 그래프(empty graph)일 때는 dmax(G)는 0으로 정의 된다.

주어진 유향 그래프, G에서 G-V가 사이클을 포함하지 않고 dmax(G-V)가 주어진 수 d보다 작거나 같게 하는 노드 최소 집합을 정하는 문제를 본 논문에서는 dmax 문제라고 한다. G-V는 그래프 G에서 노드 집합 V와 관련 에지(incident edge)를 모두 제거한 그래프이다. dmax 문제는 위에서 문헌 조사한 최소 귀환 노드 집합 문제를 일반화 한 것이다. 최소 귀환 노드 집합 문제가 NP-complete이므로 dmax문제는 NP-hard 문제임을 알 수 있다. 본 논문에서는 dmax 문제를 풀기 위한 두 가지 방법을 제시한다. 제 2절에서는 모든 가능한 탐색 영역(search area)을 나열하는 방법에 기초한 최적 알고리즘(optimal algorithm)을 소개한다. 필요한 계산 시간과 메모리를 최대한 줄이기 위한 방법이 주로 기술된다. 크고 복잡한 그래프에서는 이 방법이 효율적이지 않으므로 제 3절에서는 근사적인 알고리즘(hueristic algorithm)으로 우선 그래프가 사이클을 포함하지 않게 만들고, 그 후에 경로의 길이를 줄이는 2 단계 알고리즘을 기술한다. 제 4절에서 실험의 결과를 제시하고 제 5절에서 연구의 결과를 간단히 요약 한다.

II. 최적 알고리즘

이 절에서는 dmax 문제를 풀기 위한 최적 알고리즘을 기술한다. 여기서 최적 알고리즘이라 함은 경로의 길이를 제한하는 최소의 집합을 구하는 알고리즘을 말한다. 본 절에서 소개하는 알고리즘의 기본적인 생각은 길이가 d+1 이하인 모든 경로들을 나열하여 각 경로상의 노드를 하나 이상 선택하여 최소 집합에 포함 시키는 것이다. 이를 위하여 모든 노드로부터 가능한 모든 경로를 가로 검색 법(breath-first search)을 사용하여 생성한다. 경로를 생성하는 동안 최소 사이클을 발견한다. 최소 사이클은 다음과 같이 정의 된다.

정의 4. 사이클 A의 노드 집합이 사이클 B의 노드 집합을 포함할 경우 사이클 A가 사이클 B를 포함한다고 말한다. 최소 사이클은 그 자신을 제외한 다른 사이클을 포함하지 않는 사이클을 말한다.

최소 사이클은 깊이 우선 탐색법을 사용하여 경로를 확장하는 동안 어떤 노드의 재 방문이 이루어 질 때

발견된다. 또한 확장 된 경로가 지금까지 발견 된 어떤 사이클을 포함할 때 경로의 확장을 멈출 수 있다. 경로와 사이클의 포함 관계는 다음에 정의 되어 있다.

정의 5. 경로 A의 노드 집합이 사이클 B의 노드 집합을 포함하면 경로 A는 사이클 B를 포함한다고 말한다.

길이가 d+1인 모든 경로들과 위에 기술한 최소 사이클들을 가로 검색을 사용하여 모두 찾은 후 이들로부터 부울 노드 커버링(boolean node covering) 문제를 만든다. d+1보다 큰 길이의 사이클은 아직 발견되지 않았으나 이러한 사이클 내에 있는 모든 노드로부터 시작 되는 모든 필요한 경로가 이미 발견되었으므로 이러한 사이클을 더 이상 찾을 필요가 없다. 부울 노드 커버링 문제는 노드 커버링 문제의 확장된 문제이다[3]. 부울 노드 커버링 문제를 풀기 위하여서 노드 순서 열의 집합에서 이들 순서열 중 적어도 하나의 노드를 포함하는 최소한의 집합을 구하면 된다. 이 문제 또한 NP-complete 문제로 다항식 알고리즘이 존재하지 않는다. 최적 알고리즘 절차의 윤곽은 그림 1에서 주어 진다.

```
level=0
initialize paths
while (level<d) do{
  for every path P {
    expand it into paths
    through all outgoing edges
    for each resulting path {
     if (terminal node is starting
  node) enter in the cycle
   set
     else if (terminal node is
 in the path P) discard it
    }
    delete all paths whith cover
 any cycle found so far
  }
  increase level
}
Construct and solve
 the constraint equation
```

그림 1. 최적 알고리즘
Fig. 1. Optimal Algorithm.

이 알고리즘에는 두 가지의 계산상의 문제가 있다. 하나는 길이가 d+1인 모든 경로를 생성하는 것으로 계산에 많은 시간과 메모리가 소요된다. 다른 하나는 최소 부울커버링을 찾는 것으로 역시 계산이 복잡하다. 본 연구에서는 최소 커버를 찾기 위하여 분기 한정법(branch and bound method)을 사용하였다. 우리는 이 방법을 작은 S-그래프에 적용하였다. 대용량 고속 컴퓨터를 사용하여 더 큰 S-그래프에 이 방법을 적용할 수 있었다. 그러나, 이 논문에서 보고된 모든 결과는 Sun Sparc 워크스테이션에서 수 분 이내에 얻을 수 있는 것으로 제한 하였다. 계산 시간이 지수적으로 증가 하기 때문에 본절에서 소개된 방법은 큰 그래프에는 적용 할 수 없으므로 다음 절에서 소개할 근사적인 2 단계 방법을 연구하게 되었다.

III. 근사 알고리즘

본 논문에서 제안하는 근사 알고리즘은 두 단계로 구성되어 있다. 첫째는, 그래프에서 사이클을 제거하기 위한 최소 귀환 노드 집합을 발견하고, 둘째로 계산된 사이클이 없는 그래프의 경로의 길이를 원하는 만큼 줄인다. 최소 귀환 노드 집합 문제는 그 자체로 매우 중요하고 본 논문에서 제안하는 근사 알고리즘은 다음 소개하는 최소 귀환 노드 집합 계산 알고리즘을 계산 알고리즘의 일부로 중용 한다.

1. 최소 귀환 노드 집합 결정

임의의 그래프를 사이클이 없는 그래프로 줄이기 위한 최소 귀환 노드 집합을 찾기 위하여 그래프 리덕션 오퍼레이션(graph reduction operation)을 사용한다. 먼저 그래프 리덕션 오퍼레이션을 설명하고 그래프 리덕션 과정이 그래프를 완전히 처리 할 수 없을 때를 위한 여러 가지 근사적인 방법도 설명한다. 마지막으로 귀환 노드 집합에서 얼마의 여분의 노드를 제거 할 수 있는 근사적인 후처리 과정을 설명한다.

1) 리덕션 방법

그래프 리덕션 오퍼레이션은 최소 귀환 노드 집합을 발견할 수 있는 모든 그래프 특성을 유지하면서 그래프를 축약한다. 다섯 가지 그래프 리덕션 오퍼레이션은 다음과 같다.

① 소스 오퍼레이션 (source operation): 입력 에지가 없는 노드 v와 노드 v로부터의 모든 출력 에지를 제거한다.

② 싱크 오퍼레이션(sink operation): 출력 에지가 없는 노드 v와 노드 v로 향하는 모든 입력 에지를 제거한다.

③ 유일 입력 오퍼레이션 (unit-in operation): 입력 에지가 하나 있고 셀프 루프(self loop)가 없는 노드 v와 그것의 유일한 프리디세서 노드(predecessor node)를 하나의 노드로 묶는다.

④ 유일 출력 오퍼레이션 (unit-out operation): 출력 에지가 하나 있고 셀프 루프가 없는 주어진 노드 v에서 v와 그의 유일한 석세서 노드(successor node)를 하나의 노드로 묶는다.
⑤ 셀프 루프 오퍼레이션 (self-loop operation): 셀프 루프를 가진 노드와 그의 입출력 에지를 제거한다.

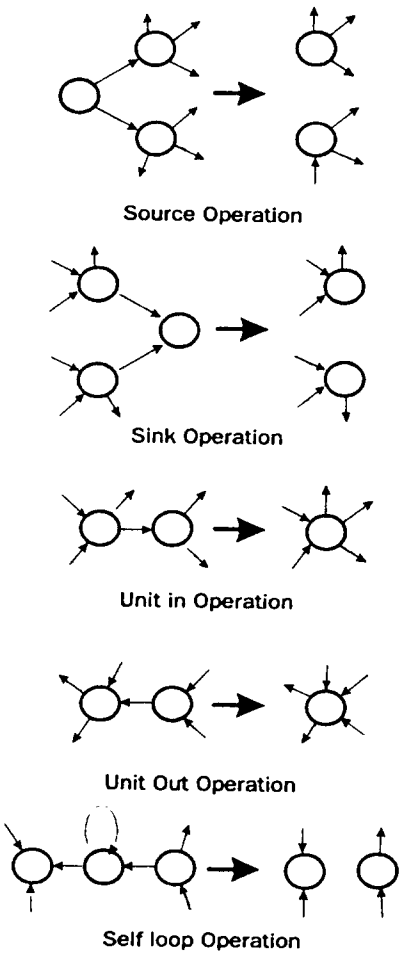

그림 2. 다섯 가지 그래프 리덕션 오퍼레이션
Fig. 2. 5 Graph Reduction Operation.

이들 5 가지의 오퍼레이션은 두 가지 클래스로 나뉘어진다. 셀프 루우프 오퍼레이션은 한 노드를 최소 귀환 노드 집합에 포함시키고 나머지 네가지 오퍼레이션은 한 노드를 최소 귀환 노드 집합에 포함시킬 고려 대상에서 제외하는 것이다. 이 과정은 항상 바른 결정을 하며 지금까지 한 결정으로부터 생긴 모든 정보를 사용한다. 따라서 이과정에서 모든 노드를 처리할 수 있으면 항상 최소 귀환 노드 집합을 발견할 수 있다.이들 오퍼레이션으로 최소 귀환 노드 집합을 발견하는 알고리즘을 구현하는 데 있어서, 위 오퍼레이션을 적용 조건을 만족하는 노드를 효과적으로 발견하는 것이 매우 중요하다. 본 연구에서는 간단한 메커니즘으로 이 문제를 해결하였다. 즉, 큐(queue)를 사용하여 리덕션 오퍼레이션을 한 노드에 적용할 때 다른 노드가 (입/출력 오퍼레이션의 경우는 결과 노드) 다음 리덕션 과정에서 사용 될지를 검사하고 큐에 넣는다. 예를 들면, 소스 오퍼레이션은 큐로부터 노드를 가지고 온 후 그것이 소스 노드인지를 조사하고 소스 노드라면 그 것을 없앤다. 소스 노드가 아니라면 큐에 다시 넣고 정의된 다른 오퍼레이션을 나중에 적용 시킬 수 있게한다. 계속해서 모든 이어지는 노드를 조사한다. 만약 다음 노드도 역시 소스 노드라면 소스 오퍼레이션을 다시 사용한다. 그렇지 않다면 그것을 큐 안에 넣는다. 모든 다른 오퍼레이션도 비슷하게 구현 된다. 셀프 루프 오퍼레이션에서는 최소 귀환 노드 집합에 제거된 노드를 더한다. 전체 리덕션 과정은 그림 3 에서 보여진다.

```
initialize queue
while(graph is not empty){
   while(queue not empty) {
         pick a node from the queue
         and apply an appropriate
         operation
   }
  if graph is not empty {
   apply heuristic
         selection to delete a node
   }
}
```

그림 3. 그래프 리덕션 방법의 개요
Fig. 3. Outline of Graph Reduction Process.

2) 근사적 선택

그림 3에서 보인 리덕션 과정이 그래프를 완벽하게 처리할 수 있다면 이 과정으로는 최적의 해를 찾을 수 있다. 이 리덕션 과정이 그래프를 완전히 처리 할 수 있는 그래프의 클래스는 Lloyd 와 Soffa에 의하여 두 방향 리덕션 가능 클래스(two way reducible class)와 일치한다[2]. 그러나 이 과정은 일반 그래프를 모두 완전히 처리하지는 못한다. 이 경우를 위하여 본 연구에서는 몇가지 근사적 선택 방법을 적용하여 남은 노드로부터 임의로 한 노드를 선택 하여 이 노드를 최소 귀환 노드 집합에 첨가하고 그래프로부터 제거한다. 가장 간단한 지역적인 방법으로 남아있는 노드들 중에

서 입력 에지의 수와 출력 에지의 최고 합이나 곱을 가진 노드를 선택하는 것이다. 또한 선형적인 비용이 드는 방법으로 그래프에서 토큰의 흐름을 시뮬레이트 하는 방법을 연구하였다. 이 방법은 각 각의 노드와 에지에 고정 된 수의 토큰을 할당한 후 그래프 상에서 토큰 흐름을 시뮬레이션한다. 많은 사이클에 접한 노드는 그들로부터 보내진 토큰을 가지는 경향이 있다. 이 통계에 바탕을 두고, 우리는 최소 귀환 노드 집합을 포함하는 노드를 선택 할 수 있다. 결과는 다음 절에서 보인다.

3) 후처리

전술한 알고리즘에서 그래프 리덕션 오퍼레이션이 더 이상 적용될 수 없을 경우 임의로 노드를 선택한다. 이러한 임의의 선택이 포함된 알고리즘으로서는 최적의 최소 귀환 노드 집합을 찾을 수 없다. 위 알고리즘의 적용 결과로 발견한 노드 집합을 면밀히 조사한 결과 간단한 후처리로 귀환 노드 집합의 크기가 줄 일 수 있음을 발견하였다. 문헌 조사 결과 이와 같은 목적을 위한 선형 시간 방법이 Rosen[17]에 의해 제안 되었음을 발견하였다. 그는 전통적인 토폴로지 정렬(topological sort) 알고리즘을 후처리 방법으로 도입하였다. 사이클이 없는 그래프의 경우와는 다르게 그래프의 사이클 때문에 토폴로지 정렬 알고리즘은 전체 그래프를 다 처리하기 전에 큐가 빈 상태를 만난다. 이 때 알고리즘은 입력 귀환 노드 집합을 읽으며 이제 까지 큐우에 들어간 적이 없는 노드를 찾아 출력 최소 귀환 집합에 포함 시키고 큐우에 넣는다. 모든 노드가 처리되고 빈 큐우에 도달할 때에 이제까지 출력 최소 귀환 집합에 포함된 노드가 최종 최소 귀환 집합이 된다. 이 방법을 잘 프로그램하면 선형적으로 실현할 수 있다.

본 연구에서 Rosen의 방법을 그래프 리덕션 알고리즘의 결과에 적용하였으나 큰 효과를 보지 못하였다. 따라서 본 연구에서는 간단하지만 뚜렷이 효과적인 방법을 선택하여 사용하였다. 이 방법은 최소 귀환 노드 집합안의 모든 노드들을 그래프로부터 지우고 하나씩 하나씩 그것이 사이클을 이루는데 이바지하는가 조사한다. 사이클을 구성하면 최소 귀환 노드 집합 안에 넣고 그래프로부터 지운다. 이로서 모든 귀환 집합에 속한 노드를 처리하여 최소 귀환 노드 집합을 구한다. 이 방법의 계산 시간 Rosen의 방법보다 크지만 최소 귀환 노드 집합 크기를 아주 많이 줄이는 것을 발견하였다.

4) 최적성

본 절에서 논한 최소 귀환 노드 집합 결정을 위한 근사 알고리즘의 효율성을 검토 하기 위해서 먼저 Smith [18]가 제안한 최적 알고리즘을 연구 하였다. 크기의 매우 크지 않거나 구조가 간단한 S-그래프의 경우 Smith의 알고리즘을 사용하여 수 분 내지 수 시간 이내에 최적의 해를 구 할 수 있다. 크고 복잡한 그래프는 많은 시간이 걸리는데 이 어려움을 해결하기 위해서 하한(lower bound)을 찾는 방법을 도입하였다. 하한은 본 절에서 논의된 알고리즘을 사용한 결과와 가까우면 본 절에서 제안된 알고리즘이 효과적임을 확신할 수 있다.

최적 해와 근접한 최소 하한를 얻기 위해 리덕션 과정이 더 이상 진행이 불가능하여 임의의 선택을 할 경우 크기 2 의 사이클 형태의 모든 노드 쌍을 찾고 이 사이클을 구성하는 에지를 그래프로부터 제거한다. 이 두 노드 중 하나는 최소 귀환 노드 집합에 포함되어야 하므로 이 중 하나가 최소 귀환 집합에 포함 된다는 제약 조건을 얻을 수 있다. 앞서의 그래프 리덕션 과정이 결과로 남은 그래프는 많은 사이클을 갖는 경향이 있으며 빈 그래프를 얻을 때까지 이러한 과정을 반복할 수 있다. 얻어진 제약 조건 집합은 최적의 노드 커버링 알고리즘으로 풀어 구하여진 노드 집합을 최소 귀환 집합에 포함 시킨다. 비록 노드 커버링은 또다른 NP-complete 문제이지만 분기 한정법(branch and bound method)을 사용하여 대부분 그래프에서 최적해를 찾을 수 있었다. 이 결과는 뒤에 보인다.

2. 경로 길이 줄이기(Reducing path lengths)

2장에서 보인 최적 알고리즘은 일반 그래프에서 최소한의 노드를 제거하여 경로 길이를 정해진 크기 이하로 제한하기 위하여 모든 노드로부터 모든 경로를 발생시킨다. 따라서 이 알고리즘은 크고 복잡한 그래프의 경우 과도한 메모리와 CPU 시간을 필요로 한다. 다음에서는 사이클을 포함하지 않는 그래프에서 같은 문제를 위한 효과적인 소개한다. 여기서 제안하는 방법의 특징은 사이클이 없는 그래프를 변경하여 변경된 그래프에서 최소 귀환 노드 집합 문제의 해가 경로 길이를 제한 하는 문제가 되는 점이다. 따라서 3.1 절에서 매우 효과적인 개발한 최소 귀환 집합 알고리즘을 사용하면 경로 길이 제한 문제를 효과적으로 풀 수 있다. 다음에 이 알고리즘을 위한 정의와 증명들을 소개한다.

먼저 주어진 사이클이 없는 그래프 G는 아래와 같이 에지를 첨가하여 그래프 G′로 변경된다.

정의 6. 그래프 G(V,E)의 변경 그래프 G′(V′,E′)의 에지 집합 E′은 E 와{(vi,vj)| P(vi,vj)의 길이 는 d + 1}의 합집합으로 정의한다.

정의 6에서 첨가된 에지를 귀환 에지라한다.원래의 그

래프에 d보다 큰 길이의 경로가 있으면 변경된 그래프는 사이클을 포함하게 된다. 그림 4에서 점선으로 보인 에지가 d=2 일 경우 첨가된 귀환 에지이다.

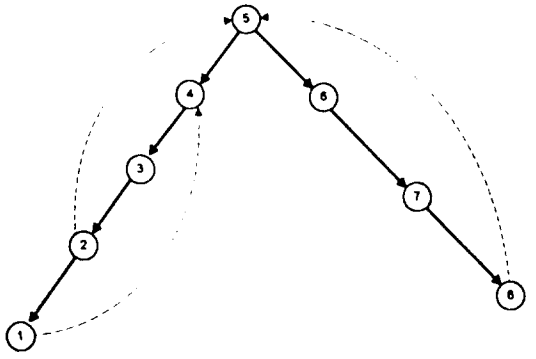

그림 4. 귀환 에지 첨가
Fig. 4. Adding feedback edges.

정리 1. 사이클이 없는 그래프 G(V,E)의 변경된 그래프 G′(V,E′)의 최소 귀환 노드 집합 V′은 d max (G-V′) ≤ d 를 만족한다.

(증명) 역을 가정하자. 그러면 G-V′는 d 보다 큰 경로 길이를 갖음을 알 수 있다. 그리하여 G′(V-V′,E′)는 적어도 하나의 사이클을 가지므로 V′이 G′의 최소 귀환 집합이 아니다.

정리 1이 암시하는 과정이 임의의 그래프, 즉 사이클을 포함한 그래프에 적용될 때 최소 귀환 집합의 크기는 dmax문제의 최적해 집합 크기보다 클 수 있다. 한 흥미 있는 점은 이 그래프 변경이 어떤 그래프 클래스에서는 dmax문제에 선형시간 최적 알고리즘을 준다는 것이다.

정리 2. 주어진 나무(tree) 구조 그래프 G의 변경된 그래프 G′의 최소 귀환 노드 집합 G′는 G의 dmax 문제의 최소의 노드 집합이다.

(증명) 정리 1로부터 충분조건은 만족된다. 필요조건을 만족되는 가를 보기 위해 변경된 그래프에서 어떤 최소 주기안에 오직 하나의 첨가된 에지가 있음을 주목하면 dmax의 문제의 해의 집합은 변경된 그래프의 모든 사이클을 자른다는 것을 알 수 있다.

정리 2의 결론으로부터 나무 구조 그래프의 dmax 문제를 위한 선형 시간 알고리즘 이 존재한다는 것을 알 수 있다. 세로 검색 방법을 이용하여 나무 구조 그래프 G를 고쳐 선형 시간 안에 변경 그래프 G′을 만들 수 있으며 변경된 그래프 G′은 흐름 그래프 리듀셔블(flow graph reducible) 클래스에 속한다[5]. Shamir는 이 클래스의 그래프를 위한 최소 귀환 노드 집합을 찾는 선형 시간 알고리즘을 발견하였다. 또한 일반적인 비주기 그래프에서도 정리 2의 반대 예제를 발견하기가 매우 힘들다. 정리 2에서 필요조건을 만족시키기 위해서는 변경된 그래프의 모든 주기를 자르는 해가 하나 존재하면 된다. 정리 2를 보다 일반적인 그래프의 클래스로 확장시키는 것이 가능하다고 믿어진다.

본 절에서 제안하는 알고리즘에서는 먼저 사이클이 없는 그래프를 만들고 그후 경로 길이를 주어진 값 d 보다 작거나 같게 제약한다. 원래의 그래프 G를 직접 변경하여 변경 그래프 G′을 만들고 최소 귀환 노드 집합을 구하는 과정을 생각할 수 있다. 그러나 실험 결과 본 절에서 제시한 2 단계 알고리즘의 경우 보다 나은 결과를 얻을 수 있었다. 주어진 그래프 G로부터 변경된 그래프 G′를 구하기 위해서 여러 노드로부터 그래프 G를 검색할 필요가 있다. 이것은 2절에서 사용한 전체 나열(exhaustive enumeration) 방식과 비슷하다. 그러나 여기에는 2개의 중요한 차이점이 있다. 먼저 첨가할 수 있는 에지의 수가 제한 되어 있다. 둘째 필요한 노드를 탐색하는 것이 아주 효율적이다. 사실상 이 변경 과정은 그래프의 재 결합 구조(reconvergent structure) 처리에 아주 효과적이다. 그래프의 재 결합 구조는 2절에서 사용한 방법이 비 효율 적이 되게 하는 주된 이유이다.

정의 8. 어떤 그래프의 모든 노드 쌍 vi , vj 에 대하여 vi 에서 vj로의 모든 경로의 노드 수가 같은 경우 그래프 G가 균형 잡혀 있다(balanced)라고 말한다.

만약 그래프가 아주 불 균형을 이루면 어떤 에지들을 여러번 검사할 필요가 있다. 이로 인한 비 효율성을 해결하기 위해 각 노드에 대해 비트 벡터(bit vector)들을 할당한다. 즉 각 비트 i는 그 노드가 깊이 i로 검사되었는지 아닌지를 나타 낸다. 만약 그래프가 균형 잡혀 있으면 오직 노드가 처음 방문될 때만 하위 노드들을 조사한다. 주어진 알고리즘의 일부를 개선하여 최소 노드의 집합을 제거하여 비주기적 그래프를 균형 있게 하는 알고리즘을 만들 수 있다. 참고문헌[13]에서는 최소의 에지를 제거하여 균형 잡힌 그래프를 만든다.

그래프 G′를 얻은 후 우리는 3.1절에서 기술한 방법으로 최소 귀환 노드 집합을 결정하여 dmax 문제의 답을 구한다.

Ⅳ. 실험결과

본 절에서는 최적 알고리즘과 근사 알고리즘을 프로그램하여 순차 논리회로로부터 추출한 S-그래프에 적용한 결과를 보인다. 여기서 사용된 순차 논리회로는 논리회로 설계, 검증, 시험 데이터 발생 등 여러 면으로 이용된 실제 회로이다[12]. S-그래프의 중요한 특

징은 참고 문헌[1]에서 보인다. 이 테이블은 각 S-그래프마다 입력의 수와 출력의 수 , 셀프 루프(self loop)의 수, 셀프 루프를 제외한 전체 에지의 수가 나타나 있다. 최소 귀환 노드 집합 결정 알고리즘은 본 논문에서 논한 그래프 문제와 깊은 관련이 있을 뿐만 아니라 그 자체로 매우 중요한 문제이므로 dmax 알고리즘의 성능을 논하기 전에 여러 종류의 최소 귀환 집합 결정 알고리즘의 성능을 실험하였다.

1. 최소 귀환 노드 집합 결정

일반 그래프(general graph)에서 본 논문에서 논한 최소 귀환 노드 집합을 결정하는 근사 알고리즘의 효율성을 검증하기 위해 이 알고리즘을 C로 구현하였다. 많은 S-그래프가 셀프 루우프를 가지고 있으므로 실험을 위하여 노드를 그래프내에 남겨둔 체 셀프 루프 에지만을 제거한 S-그래프를 주 실험 데이터로 사용하였다.

실험에서 사용된 프로그램은 먼저 Tarjan 알고리즘[11]으로 최대로 강하게 연결된 노드(strongly connected components)들의 모임을 찾는다. 이 알고리즘의 결과 어떤 사이클에도 포함되지 않는 노드들은 자신만을 포함한 집합(singleton set)에 격리돼 쉽게 제거 된다. 이러한 노드들을 그래프 리덕션 과정에서 제거하는 것 보다 이 알고리즘을 사용하여 제거하는 것이 더 효과적이다. 이 알고리즘은 선형 시간의 복잡성을 가지며 그 상수가 적어 매우 빠르다.

표 1은 사용된 S-그래프에 여러 가지 귀환 노드 집합 알고리즘을 적용한 결과를 보여준다.

3절에서 제안한 근사 알고리즘은 그래프 리덕션 연산의 적용 조건을 만족하는 노드가 존재하지 않을 때에는 적당한 노드를 선택하여 귀환 집노드 집합에 포함시키고 그 노드와 관련 에지를 그래프로부터 제거한다. 표 1에서 GS, GP, SIM은 적용된 노드 선택 방법을 나타낸다. GS에서는 입력 에지 수와 출력 에지 수의 합이 최대인 노드를 선택한다. GP에서 입력 에지 수와 출력 에지 수의 곱이 최대인 노드를 선택한다. SIM에서 토큰 패싱 시뮬레이션의 결과를 이용하여 최적 노드를 선택한다. R은 Rosen의 알고리즘을 적용한 결과를 나타낸다. Rosen의 근사 알고리즘 일반적 그래프에 적용할 수 있는 매우 빠른 알고리즘으로 알려져 있다. 이는 Shamir의 알고리즘을 약간 변형시킨 것으로써 흐름 그래프(flow graph)의 경우 최적 해를 준다. CA는 Cheng과 Agrawal의 논문에서 얻은 같은 S-그래프에서의 최소 귀환 노드 집합의 크기를 나타낸다[10]. Cheng과 Agrawal은 논리회로의 부분 스캔 시험 설계 문제를 위하여 S-그래프를 정의하고 전통적인 최소 귀환 노드 집합 알고리즘을 이용하여 해를 구하였다. 전통적인 최소 귀환 노드 집합 알고리즘의 경우 모든 사이클의 생성을 필요로 하고 많은 수행 시간을 요구한다[4]. OP/LB는 최적해의 크기나 최저 경계(lowerbound)를 나타낸다. 최적해는 Smith/ Walford 알고리즘을 이용하여 얻어지며 최저 경계는 3.1.2 절에 기술한 방법을 이용해 얻어진다. sl3207, s35932, s5378, s9234등 비교적 큰 회로를 표현하는 S-그래프의 경우 최적해를 얻을 수 있다. 왜냐하면 s35932의 S-그래프는 두방향 리듀셔블 그래프(two way reducible graph)이기 때문이다. 이들 그래프의 경우 Rosen 알고리즘도 최적 해를 줌을 알 수 있다. GS 선택 법을 사용한 경우 s13207과 s9234의 s-그래프에서 26과 36의 임의 선택이 필요로 하나 최적해가 바로 구하여 진다. s9234의 경우 GS 선택 법을 사용 하면 55 개의 노드를 갖는 해를 구할 수 있다. 그러나 2개의 노드는 후처리 과정을 통해 최소 귀환 집합으로부터 제거된다. sl423과 sl5850회로에서도 근사 알고리즘은 최저 경계와 근접한 해를 얻을 수 있다. s38417과 s38584의 경우 우리는 최적해는 물론 최저 경계도 구할 수는 없다. 그러나 GS 선택 법을 사용한 근사 알고리즘으로 최소 귀환 노드 집합을 SUN Sparc를 사용하여 10초 내에 구할 수 있다.

표 1. 최소 귀환 노드 집합의 크기
Table 1. Minimum vertex set size.

S-그래프	FF	GS	GP	SIM	R	CA	OP/LB
sl423	74	22	22	21	31	27	20
s5378	179	30	30	30	32	32	30
s9234	228	53	53	53	58	53	53
sl3207	669	59	59	59	69	61	59
sl5850	597	90	92	92	137	95	86
s35932	1728	297	297	297	297	na	297
s38417	1636	374	374	374	374	na	na
38584	1452	294	305	297	297	na	na

이같은 실험결과로 근사 알고리즘이 공학에서 나타나는 S-그래프에서 최소 귀환 노드 집합을 구하는 데 매우 효과적임을 알 수 있다. Rosen의 근사 알고리즘은 그래프가 플로우 그래프 리듀셔블하지 않을 때에 비 효율적임을 알 수 있다. Cheng과 Agrawal의 결과는 s5398, sl3207, sl423, sl5850등의 S-그래프의 경우 최적치와 큰 거리가 있음을 알 수 있다. 여기서 인용한 수는 그들의 논문에서 기록된 결과를 교정하여 우리에게 알려 준 것이다.

표 2는 사용된 S-그래프 중 비교적 큰 그래프들에 GS 근사 알고리즘을 수행할 때의 걸린 시간을 보여준다. 나타난 시간은 SUN Sparc 10 상에서 사용자 시간을 나타낸다. 후 처리 과정에 걸린 시간은 괄호 안

에 주어졌다. 실행 시간은 노드와 에지의 수와 같은 그래프 특성에 따라 달라 진다. 예를 들어 s13207보다 s15850이 많은 에지를 가지므로 더 긴 수행 시간이 소요 된다. 그러나 에지의 수가 처리 시간을 결정 짓는 유일한 요소는 아니다. s38417의 S-그래프는 s38584의 S-그래프보다 2배 많은 에지를 가진다. 그러나 후자가 전자 보다 훨씬 긴 처리 시간을 가지는 데 이는 후자의 그래프 구조가 훨씬 복잡하기 때문이다. 예상되는 후 처리 시간은 에지의 수와 깊은 관계 있다.

표 2. GS 알고리즘 수행 시간
Table 2. GS Algorithm's Execution Time.

S-그래프	s1423	s5378	s13207	s15850	s35932	s38417	s38584
sec's	0.25 (0.4)	0.21 (0.34)	0.6 (0.79)	2.9 (3.77)	0.69 (2.84)	3.58 (7.49)	6.74 (4.75)

2. 경로 길이 줄이기

일반 그래프에서 dmax 집합을 구하기 위하여 2 절에서 소개한 최적 알고리즘과 3 절에서 소개한 근사 알고리즘을 C 언어로 구현하여 실험하였다. 경로 줄이기 문제를 근사 해를 구하기 위해서는 위에서 얻은 최소 귀환 노드 집합을 사용하여 얻은 사이클이 없는 그래프에 3.2 절에서 기술한 백 에지(back edge)를 더하는 과정을 실행한다. 그 결과 그래프에 최소 귀환 노드 집합 알고리즘을 다시 적용하여 경로의 길이를 제한하였다.

먼저 근사 알고리즘이 효과적임을 입증하기 위하여 2 절에서 기술한 최적 알고리즘과 비교하였다. 표 3은 최적 결과와 실험 결과를 비교한다. 최적 알고리즘으로 얻은 dmax 집합의 크기는 괄호 안에 표시했다. 회로 s420과 s526에서 결과는 같다. 회로 s838에서 근사 알고리즘의 결과는 최적 알고리즘의 결과보다 다소 떨어 진다. 그러나 dmax = 6 일 때 최적 방법으로 dmax 집합의 크기를 구하는 것은 불가능하였다.

표 3. 근사/최적 알고리즘의 비교
Table 3. Heuristic algorithm vs. Optimal algorithm

S-그래프	d=4	d=5	d=6
s420	8(8)	7(7)	6(6)
s526	8(8)	8(8)	7(7)
s838	20(16)	17(15)	16(na)

위의 실험에서 제안 된 근사 알고리즘이 dmax 문제에 효과적임을 알 수 있다. 다음으로 이 근사 알고리즘의 큰 S-그래프에서의 효율을 실험하였다. 최소 귀환 노드 집합이 지워진 후의 S-그래프가 긴 경로를 가진 S-그래프가 실험 데이터로 선택되었다. 표 4는 큰 S-그래프에 백 에지를 첨가하는 데 걸리는 시간을 보여준다. 백 에지를 더하기 전에 S-그래프에서 최소 귀환 노드 집합에 속하는 노드와 관련 에지(incident edge)들은 제거되었다. 계산 시간은 모두 1분이 내이고 모든 그래프에서 dmax가 증가함에 따라 계산 시간이 천천히 증가함을 알 수 있었다.

표 4. 수행 시간 (d = 6)
Table 4. Execution Time (d = 6).

S-그래프	s1423	s5378	s9234	s13207	s15850	s35932	s38417
Sec's	0.1	0.1	0.4	0.8	7.6	18.9	10.6

표 5에서 이들 S-그래프에 근사 알고리즘을 적용하여 얻은 결과를 보여준다. S-그래프 s35932에서 경로 길이를 8로 줄이기 위하여 선택 노드 집합의 크기는 최소 귀환 노드 집합의 크기보다 14% 증가한다. S-그래프 s1423과 s13207의 경우 경로 길이를 8로 줄이기 위하여 약 100% 이상의 노드가 더 선택되었다. 즉 경로 길이를 8로 줄이기 위해 S-표 5 그래프에서 전체 노드 수의 18 ~ 58 %가 선택이 되었다.

표 5. dmax 집합크기
Table 5. dmax Set size.

S-그래프	비주기	d=4	d=5	d=6	d=7	d=8
s1423	22(24)	51	48	45	43	43
s5378	30(53)	41	40	39	39	38
s9234	53(31)	109	99	94	92	88
s13207	59(33)	153	143	128	120	120
s15850	90(59)	252	240	234	228	221
s35939	297(31)	415	390	365	341	339
s38417	374(27)	766	726	671	556	539

V. 결 론

본 논문은 사이클이 있는 일반 그래프에서 최소한의 노드를 제거하여 사이클이 없고 최대 경로의 길이가 주어진 값 이하로 제한되는 최소의 노드의 집합을 구하는 문제와 이 문제를 위한 알고리즘을 소개하였다.

우리는 먼저 작거나 구조가 간단한 그래프의 경우를 위한 최적 알고리즘을 제안하였다. 이 방법이 크고 복잡한 그래프의 경우 비효율적이므로 큰 그래프 문제를 위한 2 단계 알고리즘을 기술하였다. 실험의 결과로 작은 그래프의 경우 2 단계 근사 알고리즘의 결과가 최적 알고리즘의 결과에 근접하였다. 근사 알고리즘은 큰 그래프에 적용되어 시간적으로 효과적임이 입증되었다. 큰 그래프의 경우 근사 알고리즘의 최적성은 더 많은 연구를 필요로 한다.

아울러 그래프 리덕션에 기초한 최소 귀환 노드 집합 알고리즘의 효율성을 실험적으로 확인하였다. 최소 귀환 노드 집합 문제는 아주 어려운 것으로 알려져 있지만 그래프 리덕션에 기초한 간단한 알고리즘이 효율

적으로 최적에 가까운 최소 귀환 노드 집합을 얻는 것을 보였다. 간단한 후처리 과정이 사용돼 근사 최소 귀환 집합 알고리즘의 효율성을 개선하였다.

참 고 문 헌

[1] D. H. Lee, Test Generation and Partial Scan Designs for Synchronous Sequential Circuits, Ph.D. Thesis, University of Iowa, July 1992.

[2] E.L.Lloyd and M.L.Soffa,"On Locating Minimum Feedback, Vertex Sets," *Journal of Algorithms* vol.9, pp.470-493, 1988.

[3] R.M.Karp,"Reducibility between combinational problems," Complexity of Computer Computations, R.E.Miller and J.W.Thatcher, Eds. New York, Plenum Press, pp. 85-103, 1972.

[4] Tierman,J.C. "An Efficient search algorithm to find the elementry circuits of a graph," Comm. ACM 13, vol.12, pp. 722-727, 1970.

[5] A.Shamir, "A Linear Time Algorithm for Finding Minimum Cutsets in Reducible graphs," SIAMJ. Computing Vol. 8, No.4, pp.645-655, November 1979.

[6] C.Wang,E.L.Lloyd, and M.L.Soffa, "Feedback Vertex Sets and Cyclically Redudicible Graphs," *Journal of the Association for Computing Machinery*, vol.32, No.2, pp.296-313, April 1985.

[7] B.K.Rosen, "Robust Linear Algorithm for Cutsets," *Journal of Algorithms* vol.3, pp. 205-217, 1982.

[8] G.W.Smith and R.B.Walford, "The Identification of a Minimal Feedback Vertex Set of a Directed Graph," IEEE Transactions on Circuits and Systems, vol. CAS-22, No.1 January 1975.

[9] H.Levy and D.W.Low, "A Contraction Algorithm for Finding Small Cycle Cutsets," *Journal of Algorithms* vol.9, pp.470-493, 1988.

[10] K.T.Cheng and V.D.Agrawal, "A Partial Scan Method for Sequential Circuits with feedback," IEEE Transactions on Computers, vol.39, NO.4, pp.544-548, April 1990.

[11] R,Tarjan,"Depth-first search and linear graph algorithms," *SIAM J. Computing*, vol.1, pp.146-160, 1972.

[12] F.Brglez,D.Bryan,and K.Kozminski, "Cobinaional Profiles of Sequential Benchmark Circuits," Proc. of International Symposium on Circuits and Systems,pp. 1929-1934, May 1989.

[13] R,Gupta,R.Gupta and M.A.Breuer, "BALLAST: A Methodolgy for Partial Scan Design," Digest of Papers of the 9th.

[14] A.Miczo,Digital Logic Testing and Simulator,Harper & Row, Publishers,Inc., pp. 98-115, 1986.

저 자 소 개

李 東 浩(正會員)

1956년 생. 1979년 2월 서울대학교 전자공학과 졸업. 1981년 2월 KAIST 전산학과 졸업. (이학석사) 1981년 3월 ~ 1982년 7월 KAIST 전산연구소 기술원. 1982년 8월 ~ 1992년 7월 ETRI 선임연구원. 1992년 8월 (미) 아이오와 대학 전산학과 졸업 (박사). 1992년 8월 ~ 1993년 2월 (미) Motorolar senior CAD Engineer 1993년 3월 ~ 현재 경북대학교 전자공학과 전임강사