점진적 지배자연결그래프에 관한 알고리즘

심손권*, 유희종*, 신현덕*, 이대식*, 장재춘**, 안희학*
*관동대학교 컴퓨터공학과
**강릉영동대학 컴퓨터정보계열
e-mail:sksim@kwandong.ac.kr

An Algorithms on Incremental Dominator-Join Graph

Son-Kweon Sim*, Heui-Jong Yu*, Hyun-Deok Shin*,
Dae-Sik Lee*, Jae-Chun Jang**, Heui-Hak Ahn*
*Dept of Computer Engineering, Kwandong University
**Dept of Computer & Information, Yeongdong College

요 약

객체지향 프로그래밍 방식으로 인해 프로그램의 재 사용성이 증대되었다. 객체들을 재 사용함으로서 프로그램을 수정 갱신하는 일이 더욱 많아져 점진적 데이터 흐름 분석 기법은 코드 최적화의 성능을 향상하는 중요한 방안이 되었다. 이에 본 논문에서는 데이터 흐름 분석을 위한 지배자연결그래프를 점진적으로 구성하는 알고리즘을 제안하고 이의 타당성과 성능을 실험을 통하여 제시하였다.

1. 서론

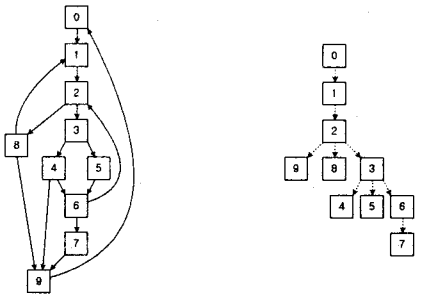
코드 최적화 방안의 기본이 되는 데이터 흐름 분석을 효과적으로 처리하기 위하여 본 논문에서는 점진적 지배자연결그래프 구성 알고리즘을 제안하고 실험을 통하여 알고리즘의 타당성과 성능을 측정하였다.

알고리즘의 타당성은 기존의 방법으로 구성한 지배자연결그래프와 점진적으로 구성한 지배자연결그래프를 비교하여 알고리즘의 타당성을 증명하였고, 점진적 지배자연결그래프 구성 방법은 기존 알고리즘에 비해 지배자 계산과 연결가지 정보 테이블의 재구성 단계를 생략시켜 수행시간을 단축시킨다.

2. 지배자연결그래프

흐름 그래프는 단순히 프로그램의 제어 구조나 데이터의 흐름만을 알 수 있다. 또한 지배자 트리는 프로그램의 지배 구조만을 나타내기 때문에 데이터의 전체 흐름을 파악할 수가 없다[1][3].

데이터의 흐름을 효과적으로 분석하기 위하여 Sreedhar는 흐름 그래프와 지배자 트리의 장점만을 취하여 지배자연결그래프를 제안하였다[4-8]. 지배자연결그래프는 흐름그래프의 노드 정보와 가지 정보, 그리고 지배 정보를 종합하여 만들어진다. 지배자연결그래프를 생성하기 위하여 흐름그래프의 각 노드에서 가지 정보를 추출한 다음 지배자 계산을 하여 지배자 가지 정보를 추출한다. 흐름그래프 가지 정보에서 지배자 가지 정보를 빼면 간단하게 연결가지 정보를 얻을 수 있다. 또한 지배자 특성 중 피지배 노드는 지배 노드의 레벨+1이므로 각 노드의 레벨 정보를 얻을 수 있다.



[그림 1] 흐름그래프와 지배자트리

[그림 1]은 흐름그래프와 그에 따른 지배자트리를 나타내었다. <표 1>은 [그림 1]에 대한 연결가지 정보 테이블이다[9].

<표 1> 연결가지 정보 테이블

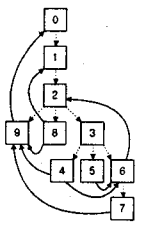
흐름 그래프		지배자 트리		레벨정보	연결가지
시작노드	도착노드	시작노드	도착노드		
노드 0	1	노드 0	1	레벨 0	null
노드 1	2	노드 1	2	레벨 1	null
노드 2	3, 8	노드 2	3, 8, 9	레벨 2	null
노드 3	4, 5	노드 3	4, 5, 6	레벨 3	null
노드 4	6, 9	노드 4	null	레벨 4	4→6, 4→9
노드 5	6	노드 5	null	레벨 4	5→6
노드 6	2, 7	노드 6	7	레벨 4	6→2
노드 7	9	노드 7	null	레벨 5	7→9
노드 8	1, 9	노드 8	null	레벨 3	8→1, 8→9
노드 9	0	노드 9	null	레벨 3	9→0

지배자트리와 연결가지 정보 테이블의 연결가지 정보를 입력받아 지배자연결그래프를 생성하는 알고리즘을 나타내면 [알고리즘 1]과 같다.

[알고리즘 1]은 지배자트리의 정보와, 연결가지 정보를 입력받아 연결가지의 도착노드를 지배자 노드의 연결가지 정보에 배정하여 지배자트리에 연결가지를 생성한다.

```
procedure make_dj(dom_tree, join_edge);
begin                    /*지배자 트리와 연결가지*/
  while join_edge do
      dom_node.jedge := reach_node
end;                          /*연결가지 생성*/
```

[알고리즘 1] 지배자연결그래프 생성 알고리즘



[그림 2] 지배자연결그래프

[그림 1]의 지배자트리는 [알고리즘 1]에 의하여 [그림 2]의 지배자연결그래프로 구성된다. [그림 2]의 지배자연결그래프에서 점선은 지배자가지를 뜻하며, 실선은 연결가지를 의미한다.

지배자연결그래프는 흐름그래프의 모든 정보와 지배자트리의 정보를 합하여 만들어지기 때문에 지배자트리로 만들어지지 않는 비순차적 프로그램에 대해서도 데이터 흐름을 분석할 수가 있다[2].

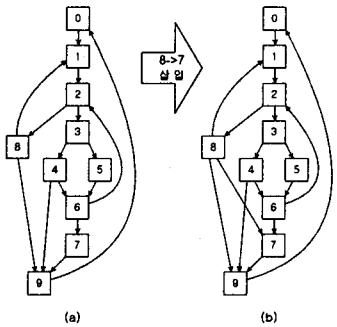
3. 점진적 지배자연결그래프 구성

점진적 지배자연결그래프 구성은 프로그램의 수정과 갱신으로 인해 변경된 프로그램에 대하여 지배자연결그래프를 새롭게 구성하지 않고 기존의 지배자연결그래프에서 변경된 부분에 대하여만 수정하여 재 사용하는 방법이다.

3.1 연결가지 삽입 시 점진적 구성 알고리즘

연결가지 삽입 시 점진적 구성 알고리즘은 프로그램 소스코드 수정시 일부분의 소스코드가 추가되었을 경우에 지배자연결그래프를 점진적으로 구성하는 알고리즘이다.

[그림 1] 흐름 그래프의 8번 노드에서 7번 노드로 가는 가지가 삽입된 경우를 나타내면 [그림 3]과 같다.



[그림 3] 연결가지 삽입

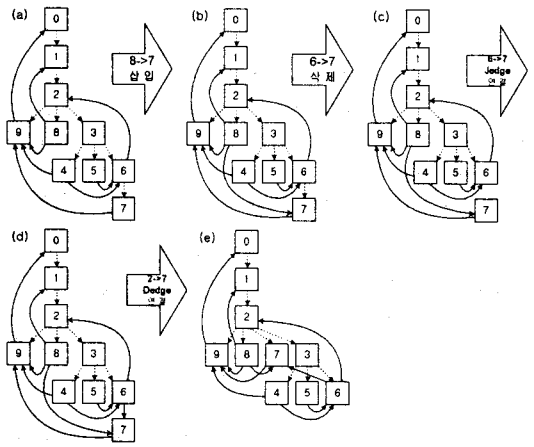
[알고리즘 2]는 가지 8→7의 삽입으로 인하여 영향 받은 노드들에 대하여 점진적으로 지배자연결그래프를 구성하기 위한 알고리즘이다.

```
Procedure Update_InsertEdge(x, y)
begin
  nca := nc_ancestor(x, y) ;
  idf := Dom_Frontier(y) ;
  while idf do
    begin
      if idf.level > nca.level + 1 then
        affected_list := idf
    end ;
  if nca <> x then
    x.jedge := y ;
  while affected_list do
    begin
      idom := affected_node.dom ;
      affected_node.dom := null ;
      if idom.child = affected_node then
        idom.jedge := affected_node ;
      nca.dom := affected_node
    end ;
  while affected_list do
    affected_node.level := affected_node.dom + 1
end ;
```

[알고리즘 2] 연결가지 삽입 시 점진적 지배자연결그래프 구성 알고리즘

흐름 그래프에 삽입된 가지 8→7에 대해 [알고리즘 2]를 적용하면, 가지 8→7의 공통의 조상은 2번 노드이고, 7번 노드의 지배영역에 있는 노드는 9번 노드이다. 하지만 9번 노드는 2번 노드의 레벨+1과 같으므로 7번 노드만이 연결가지 삽입으로 영향 받은 노드가 된다. 삽입된 가지의 시작노드와 공통 조상 노드가 같지 않으므로 8번 노드와 7번 노드를 연결가지로 연결한다. 영향 받은 7번 노드에 대하여 삽입되기 전의 지배자 가지를 삭제한 후, 가지 6→7은 흐름그래프 가지이므로 연결가지로 연결한다. 그리고 가지 8→7의 공통 조상과 영향 받은 7번 노드를 지배자 가지로 연결한다. 마지막으로 영향 받은 7번 노드의 레벨을 공통 조상의 자식노드로 갱신한다.

[알고리즘 2]에 의해 [그림 2]의 지배자연결그래프는 [그림 4]와 같이 점진적으로 구성된다.



[그림 4] 가지 삽입에 의한 점진적 지배자연결그래프 구성

[그림 4]에서 삽입된 가지의 시작노드와 공통의 조상노드가 다르므로 노드 8과 노드 7을 연결가지로 연결하고, 영향 받은 노드 7을 지배하고 있는 노드 6은 더 이상 노드 7을 지배하지 못하므로 지배자 가지를 삭제한다.

[그림 4]의 (c)에서 노드 6과 노드 7은 흐름 그래프의 가지로 연결되어 있으므로 노드 6과 노드 7을 연결가지로 연결한다. 최종적으로 영향 받은 노드 7을 새로운 지배자 노드 2와 지배자 가지로 연결한다. 따라서 노드 7은 노드 3, 8, 9 와 같은 레벨로 끌어올려 진다.

3.2 연결가지 삭제 시 점진적 구성 알고리즘

연결가지가 삭제될 때 점진적 지배자연결그래프 구성 알고리즘은 소스코드 수정시 일부분의 소스가 삭제되었을 경우에 대한 점진적 알고리즘이다. 소스코드의 일부 수정은 흐름 그래프의 가지 삭제로 대신할 수 있다.

제안한 점진적 지배자연결그래프 구성 알고리즘에 대한 실험은 흐름 그래프에 가지가 삭제되었을 경우 가지 삭제로 인해 영향 받은 노드를 찾고 영향 받은 노드들의 지배자를 재구성하여 데이터 흐름을 분석하는 실험이다.

[그림 3]의 (b)에서 (a)로 8→7의 가지를 삭제하여 보자. [알고리즘 3]은 가지 8→7의 삭제로 인하여 영향 받은 노드들을 찾고 점진적으로 지배자연결그래프를 구성하는 알고리즘이다.

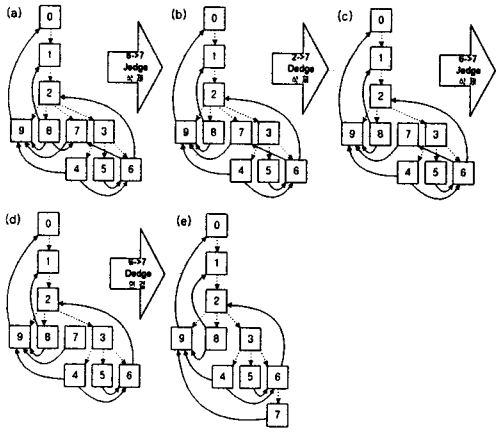
```
Procedure Update_DeleteEdge(x, y)
begin
  idf := Dom_Frontier(y) ;
  while idf do begin
       if idf.level = y.level then
          affected_list := idf
  end ;
  idom := y.dom ;
  idom_list := subtree(idom) ;
  while idom_list do
    begin
          possible_node_list := possible_affected(idom_list) ;
          pseudo_node_list := pseudo_affected(idom_list) ;
          not_node_list := not_affected(idom_list)
    end ;
  flag := true ;
  while flag do
    begin
      flag := false ;
      domtemp := idom_list ;
      while possible_node_list do
        begin
          ancestor := pred(possible_node_list) ;
          if ancestor = pseudo_node_list then
            domtemp = domtemp ∩ ancestor ;
          affected_list.dom := domtemp ;
          if idf.dom != affected_list.dom then
            flag := true
        end
    end
end ;
```

[알고리즘 3] 연결가지 삭제 시 점진적 지배자연결그래프 구성 알고리즘

흐름그래프에서 삭제된 가지에 대해 [알고리즘 3]을 적용하면, 도착노드 7의 지배에 있는 노드는 9이다. 따라서 가지 삭제로 인해 영향 받는 노드는 7과

9 이다. 노드 7의 직접 지배자 노드 2의 서브 트리에 있는 모든 노드에 대하여 영향 받는 노드(Possibly Affected Node), 영향 받을 가능성이 있는 노드(Pseudo Affected Node), 영향 받지 않는 노드(Not Affected Node) 등의 집합으로 나누어야 한다. 영향 받은 노드 7, 9의 새로운 지배자를 계산하고, 새로운 지배자의 자식으로 지배자연결그래프를 점진적으로 갱신한다.

[알고리즘 3]에 의해 점진적으로 구성된 지배자연결그래프는 [그림 5]와 같다.



[그림 5] 가지 삭제로 인한 점진적 지배자연결그래프 구성

그림에서와 같이 점진적 알고리즘에 의해 지배자연결그래프는 가지가 삽입되기 전의 지배자연결그래프와 같은 형태로 복원되었다. 이로서 점진적 지배자연결그래프 구성 알고리즘의 정확성이 증명되었다.

4. 실험 결과

본 논문에서 제안한 알고리즘의 성능을 측정하기 위하여 SUN Ultra-60 시스템에서 C 언어를 사용하였다. 실험에 사용된 흐름그래프는 각각 노드 수와 연결가지의 수가 서로 다른 흐름그래프를 예로 들었다. <표 2>는 흐름그래프 종류와 그에 따른 기존 방법과 제안한 방법과의 지배자연결그래프를 구성하는 수행속도에 대한 측정 결과이다.

<표 2> 제안한 알고리즘의 수행속도 측정 결과

	흐름그래프1	흐름그래프2	흐름그래프3	흐름그래프4	흐름그래프5
가지 수	8	11	15	15	21
노드 수	7	9	10	12	15
방법1	0.67	0.71	1.13	1.09	1.79
방법2	0.61	0.64	0.98	0.94	1.52

방법1 : 순차적인 방법에 의한 지배자연결그래프 구성 방법
방법2 : 점진적 지배자연결그래프 구성 방법
실행시간단위 : second

본 논문에서 제안한 점진적 지배자연결그래프 구성 알고리즘은 기존의 방법에 비해 약 15% 가량 수행속도를 단축시킬 수 있었다.

5. 결론

프로그램 개발 시 프로그램 수정은 빈번히 발생하므로 점진적 지배자연결그래프 구성 알고리즘은 데이터 흐름 분석을 함에 있어 기존의 방법과 비교하여 지배자 재 계산과정과 메모리 사용의 대부분을 차지하는 연결가지 정보 테이블을 구성하지 않으므로 효과적으로 메모리를 사용할 수 있게 하고, 프로그램 컴파일 시간을 줄일 수 있다.

향후 연구과제로는 점진적 지배자연결그래프 구성 알고리즘을 이용하여 데이터 흐름 분석 기법과 연결하여 데이터 흐름 분석의 성능을 향상시키는 연구가 필요하다.

참고문헌

[1] Aho, A. V., Sethi, R. and Ullman. J. D., "*Compilers Principles, Techniques, and Tools*", Addison-Wesley Publishing Co., 1986.

[2] Ramalingam, G. and Reps, T., "An Incremental Algorithm for Maintaining the Dominator Tree of a Reducible Flowgrap h", *In Proceedings of the ACM SIGPLAN-SIGACT Symposium on the Principles of Programming Languages*, pp.314-325, Jan. 1994.

[3] Ryder, B. G. and Paull, M. C., "Incremental Data-flow Analysis Algorithms", *ACM Computing Surveys*, Vol. 18, No. 3, pp.277-316, Sept. 1986.

[4] Sreedhar, V. C. and Gao, G. R., "A Linear Time Algorithm for Placing Ø-nodes", *In Proceedings of the ACM SIGPLAN-SIGACT Symposium on the Principles of Programming Languages*, pp.62-73, New York, 1995.

[5] Sreedhar, V. C. and Gao, G. R., "Computing Ø-nodes in Linear Time Using DJ Graphs", *Journal of the ACM*, Vol. 3, No. 4, pp.191-213, 1996.

[6] Sreedhar, V. C., Gao, G. R. and Lee, Y., "Identifying Loops using DJ Graphs", *ACM Transactions on Programming Languages and Systems*, Vol. 18, No. 6, pp.649-658, Nov. 1996.

[7] Sreedhar, V. C., Gao, G. R. and Lee, Y., "A New Framework for Exhaustive and Incremental Data Flow Analysis using DJ Graphs", In *Proceedings of the SIGPLAN'96 Conference on Programming Language Design and Implementation*, pp.278-290, New York, 1996.

[8] Sreedhar, V. C., Gao, G. R. and Lee, Y., "Incremental Computation of Dominator Trees", *ACM Transactions on Programming Languages and Systems*, Vol. 19, No. 2, pp.239-252, Mar. 1997.

[9] 심손권, 안희학, "경로 압축을 이용한 DJ 그래프의 지연 감축 알고리즘", *한국정보처리학회 논문지*, 제9-A권 2호, pp.171-180, 2002.