일반 트리 도시 알고리즘

허 혜정, 김 수아, 이 주영
덕성여자대학교 전산학과

A Drawing Algorithm for General Trees

Hyejung Hur, Su-Ah Kim & Ju-Young Lee
Dept. of Computer Science Duksung Women's University

요 약

트리는 실세계의 모델을 가시적으로 알기 쉽게 표현하기 위한 자료구조로서 자주 이용되어진다. 따라서 트리의 이해도와 판독성을 증가시켜 정보의 의미를 명확하게 전달하는 것은 매우 중요하다. 이를 위해서는 공간과 시각적 측면을 고려하여 적절한 노드의 위치 배정이 필요하다. Tilford는 최소면적의 공간에 트리를 보기 좋게 도시하는 알고리즘을 제시하였는데, 이는 트리의 전체적 구조가 왼쪽으로 치우칠 뿐 아니라, 도시시에 노드의 불필요한 이동이 많이 일어나는 등의 단점을 가지고 있다. 본 논문에서는 이러한 단점을 개선하여 트리를 도시하는 알고리즘을 제시한다.

1. 서 론

그래프 도시이란 주어진 그래프를 정의된 미적 기준에 따라 평면이나 공간과 같은 기하학적 대상에 포함시키거나 연관시키는 방법과 체계를 의미한다. 그래프를 보기 좋게 도시한다는 것은 그래프의 효율적인 시각화를 통하여 그래프 구조에 대한 이해도를 증진시키며 정보의 의미를 명확하게 전달할 수 있어서 판독성을 증가시킨다. 이러한 그래프 도시에 관한 연구는 도시에 필요한 면적을 최소로 하는 연구[4]와 시각적 측면과 면적을 동시에 고려하는 연구[1,2,3] 두 가지로 크게 나뉘어 그동안 끊임없이 이루어져 왔다.

본 논문에서는 실세계의 모델을 이해하기 쉽도록 가시적으로 표현하는 데 자주 이용되는 자료 구조인 트리를 후자의 방법으로 도시하는 알고리즘을 제안한다. 트리는 정렬, 탐색, 컴파일러, 결정 트리(decision tree), 조직 차트, 데이터베이스 모델, 디렉토리의 구조등 넓은 응용분야를 갖는다. 따라서 트리를 공간과 시각적 측면을 고려하여 도시하는 방법에 대한 연구는 현대 컴퓨터 응용분야에서 중요한 위치를 차지하고 있다. 트리의 시각화 작업시 노드의 위치 배정은 구조적인 효율성을 증가 시킬 수 있기 때문에 매우 중요하다. 이러한 트리 도시시에 고려해야 할 미적 제약 조건들을 Wetherell, Shannon, Tilford 그리고 Reingold가 [표-1]과 같이 제시하였다. [1,3]

미적 제약 조건을 준수하여 트리를 도시하는 알고리즘들은 수없이 많이 개발되어졌다. 최소 면적과 시각적 효과에 초점을 맞춘 Tilford가 제안한 알고리즘은 왼쪽으로부터 최소의 간격을 유지하도록 노드의 위치를 결정하기 때문에 트리의 전체적 구조가 왼쪽으로 치우칠 뿐 아니라, 도시시에 노드의 불필요한 이동이 일어난다는 단점을 가지고 있다. 이로 인해 본 논문에서는 Tilford의 알고리즘이 가지는 단점을 개선하여 불필요한 노드들의 이동횟수를 줄이는 알고리즘을 제시한다.

1. 트리의 같은 레벨(level)에 있는 노드들은 일직선상에 위치하며, 각 레벨은 서로 평행하다.
2. 이진 트리에서 왼쪽, 오른쪽 자식 노드는 각각 부모의 좌, 우에 위치한다.
3. 부모는 이들 자식들의 중앙에 위치한다.
4. 트리와 그의 거울(mirror)이미지는 서로 대칭적(reflection) 형태로 그려져야 한다.
5. 같은 모양의 서브 트리는 항상 같은 형태로 그려져야 한다.

[표-1. 트리의 미적(Aestheric) 정의]

2. Tilford의 트리 도시 알고리즘

Tilford는 [표-1]과 같은 미적 제약 조건을 만족하면서 트리를 도시하는데 필요한 면적을 최소로 하는 알고리즘을 개발하였다.

Tilford의 트리 도시 알고리즘에서는 [정의 2.1]에서 정의된 변수들이 사용되어진다.[1]

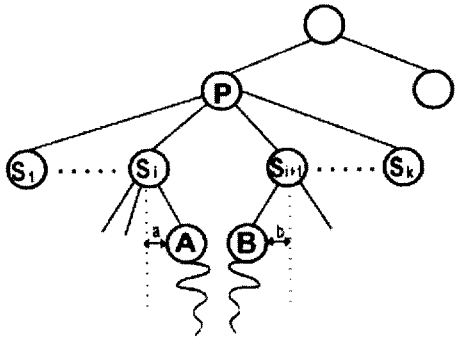
정의 2.1

cursep은 알고리즘 실행 중 현재 레벨에서 두 인접 노드 사이의 간격을 말하며, minsep은 두 노드가 최소한 유지해야 하는 간격으로 미리 정해준 값을 말한다. offset은 부모 노드에 대한 자식 노드가 가지는 상대적 간격을 말하며, sep(i)는 노드 P가 있을 때 i

번째 자식과 i+1번째 자식 사이의 간격을 말한다.

Tilford의 트리 도시 알고리즘의 기본 원칙은 후위 우선 순회(postorder traversal)에 의해 상대적 좌표를 결정한 후, 트리 노드의 간격들을 비교하여 노드의 좌표를 재귀적(recursive)으로 재조정하는 것이다. 트리 노드 좌표의 재조정 방법은 다음과 같다. [그림-1]에서 S_i와 S_{i+1}의 cursep을 minsep으로 최소 간격만큼 떨어져 있게 한 후, S_i의 마지막 자식 노드를 A라 하고, S_{i+1}의 첫 번째 자식 노드를 B라 하자. 여기서 노드 A의 S_i에 대한 상대 간격이 a라 하면, 노드 B의 S_{i+1}에 대한 상대 간격이 b라고 둔다. A와 B사이의 minsep을 유지하기 위해 S_i와 S_{i+1}의 간격, cursep을 cursep + a+b 로 변경한다. 위와 같은 과정을 서브 트리 A의 오른쪽 가장자리에 있는 노드들과 서브 트리 B의 왼쪽 가장자리에 있는 노드들을 각 레벨 별로 따라 내려가면서 두 인접 노드 사이의 간격 을 계속 조정한다. δ만큼 작으면 S_i와 S_{i+1}의 간격 cursep = cursep + δ로 변경하고 그에 따른 S_{i+1}의 서브 트리 내의 노드들의 위치도 계속적으로 변경된다. 서브 트리 A, B의 리프에 도달하면 스레드(thread)의 유무를 확인한 후 있다면 노드의 위치를 재조정한다. 마지막으로 루트(root)노드는 트리가 그려진 x좌표의 전체 영역의 중간 지점을 x좌표로 설정한다.

Tilford의 트리 도시 알고리즘은 최소 면적을 갖으면서 미적 제약 조건을 만족한다. 그러나 하위 레벨로 가면서 상위 레벨의 노드들까지 계속적으로 이동하게 되어 있어서 전반적인 노드의 이동 회수가 많으며, 왼쪽으로부터 최소의 간격을 유지하도록 노드의 위치를 변경하기 때문에 [그림-2]의 (a)와 같이 노드들이 왼쪽으로 치우치는 단점을 갖는다.



[그림-1 Tilford 알고리즘의 노드 간격 조절]

3. 일반 트리 도시 알고리즘

본 논문에서 제안하는 일반 트리 도시 알고리즘은 Tilford의 알고리즘의 단점을 개선했다. 서브 트리를 그린 후에 좌, 우의 서브 트리의 위치를 비교하면서 노드들을 이동시키는 Tilford의 트리 도시 알고리즘은 후위 우선 순회로 진행되기 때문에 큰 서브 트리로 올라갈수록 계속적으로 노드들이 이동해야한다. 이러한 노드의 이동횟수를 줄이기 위해 우리는 좀더 단순한 일반 트리 도시 알고리즘을 제안한다. 알고리즘은 다음 [표-3]과 같다. 후위 우선 순회에 의해 좌표를 결정해서 트리를 도시하며, 각 노드의 좌표는 (x, y)로 표기한다. 단, x ≥0, y ≥0 이다.

[그림-2. 트리 도시의 예]

[그림-3]에서와 같은 예를 살펴보자. [그림-3] (a)에 있는 일반 트리를 후위 우선 순회하면 b, e, i ,o ,p ,j ,k ,f ,c ,g ,l ,m ,n ,h ,d ,a와 같다. b노드를 임의의 위치 (1, 0)에 놓는다. max_y는 현재까지 그려진 노드들 중에 가장 오른쪽에 그려진 노드의 y 좌표이다. 다음 노드의 y좌표는 x가 같지 않다면, ① max_y + minsep - | max_x - x | /2 이고, x가 같다면, ② y + minsep -| max_x - x | /2이다. 즉, 다음 e 노드는 b노드와 1 레벨 차이를 갖으며, x가 서로 같지 않으므로 ①번식이 적용된다. 그리고 minsep이 1로 주어졌다면 e 노드의 좌표는 (2, 0 + 1 - 0.5) 즉, (2, 0.5)이다. 다음 노드 i의 좌표는 (3, 0.5 + 1 - 0.5) 즉 (3, 1)이어야 한다. 그러나 e 노드에서 i 노드로의 이동시 각 노드의 형제 노드 수에 따라 더 이동할 수 있는 원칙이 주어진다. e 노드의 형제는 2이고 i 노드의 형제는 3이므로 | 2-3 | / 2 만큼을 y좌표에서 마이너스 할 수 있다. 즉, 좌표는 (3, 0.5)이다. 그러나 이러한 경우에 있어서 대부분의 일반적인 경우에는 이와 같은 원칙이 성립하나 이와 같은 원칙이 성립되지 않는 부분이 있음을 고려하여 현 레벨에서 가장 오른쪽에 있는 노드와 비교하여 항상 minsep 만큼을 유지하여야 하며 그렇지 못한 경우 minsep을 유지하는 좌표로 고정한다. 다음 노드 o, p도 위에서와 마찬가지로 적용이 된다. 다음 노드 j는 o와 p 노드의 부모 노드로서 두 노드의 중간 위치에 놓이게 된다. 다음 노드 k는 ②번식을 적용해서 y좌표를 설정한다. 이와 같이 [그림-3] (a)에 있는 트리들의 그리면 [그림-3] (b)와 같이 완성된다. 이 일반 트리는 최소 간격에 미적 정의를 따라 트리를 도시 해준 것이다.

일반 트리 도시 알고리즘은 정리3.1에서 증명된 것처럼 최소의 면적에 도시 되며, 또한 처음에 제시했던 미적 정의를 따른다.

후위 우선 순회에 의해 상대적 좌표를 결정한다.
노드의 좌표는 (x, y)로 표현하며 x≥0, y≥0이다.
각 노드의 레벨을 *level*로 표기한다.
*max_y*는 현재 그려진 트리의 노드 중 y좌표가 가장 큰 노드의 좌표로 설정하며, 그 노드의 x좌표를 *max_x*로 설정한다.

[단계-1]: 후위 우선 순회에 의해 처음 방문한 노드에서,
x = *level*; y = *k*; // y는 임의의 정수(*k*)를 선택한다.
max_x = x; *max_y* = y;
[단계-2]: 다음으로 방문하는 노드가

2-1) 이전에 방문한 노드들의 부모 노드가 아닐 경우,
x = *level*;
if (*max_x* != x) y = *max_y* + *minsep* - | *max_x* - x | /2;
else y = y + *minsep* - | *max_x* - x | /2;

2-1-1) 방문한 노드가 각 서브트리의 왼쪽 노드일 경우, 현재 노드의 형제 수가 3이상이고 이전에 방문한 노드의 형제 노드의 수보다 1이상 큰 경우,
if (*max_x* > x) y = y - *diff_sibling*() /2;
설정된 좌표가 같은 레벨의 가장 오른쪽 노드의 위치 -*max_node(x)*-에 *minsep*만큼 떨어진 위치보다 작은 경우 *minsep*만큼의 위치에 좌표를 고정한다.
y = *max_node*(x) +*minsep*;
goto 단계-3

2-1) 이전에 방문한 노드들의 부모일 경우,
이전에 방문한 노드들의 가운데에 위치한다.
x = *level*; y = *middle*(*children_node*);

[단계-3]:
if (y >= *max_y*) { *max_x* = x; *max_y* = y; }
모든 노드를 방문할 때까지 단계-2를 수행한다.

[표-3. 일반 트리 도시 알고리즘]

정리 3.1

일반 트리 도시 알고리즘에 의해 도시된 일반 트리는 미적 기준을 따르면서 최소 면적을 갖는다.

증명 : 일반 트리 도시 알고리즘에 의해 도시된 일반 트리는 각 서브 트리 사이의 양끝 노드들의 간격은 minsep 이상을 유지하나 그들 중 적어도 하나는 정확히 minsep만큼의 간격을 갖는다. 따라서 각 서브 트리 사이의 간격이 하나의 minsep으로 인해 더 이상 줄어들 수 없기 때문에 최소의 면적으로 도시된다.■

정리 3.2

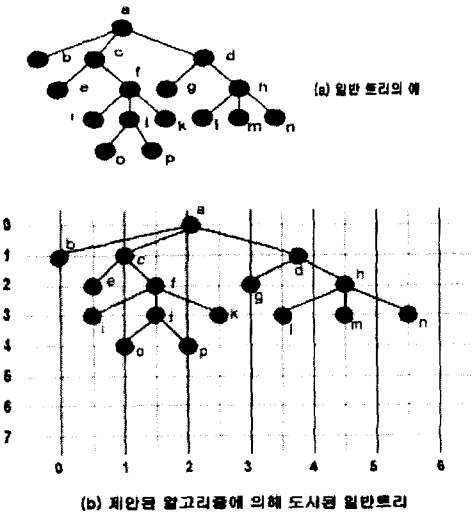
일반 트리 도시 알고리즘에 의한 도시는 트리의 미적 기준에 따른다.

증명 : 일반 트리 도시 알고리즘에 의해 도시된 트리는 그리드 상에 그려지므로 같은 레벨(level)에 있는 노드들은 일직선상에 위치하고, 각 레벨은 서로 평행하다. 부모는 자식의 중앙에 위치하며 그 자식 노드들은 부모의 좌, 우에 위치한다. 서브 트리들을 도시하는데 일정한 원칙이 적용되므로 같은 모양의 서브트리는 같은 형태로 그려지며 트리의 거울(mirror) 이미지는 대칭적으로 도시된다. 따라서 일반 트리 도시 알고리즘은 미적 기준을 따른다.■

또한 후위 우선 순회에 의해 각 좌표를 설정하고 이동하지 않는다. 따라서 이동성이 없다. 그러나 Tilford 알고리즘은 상대적 좌표를 설정한 후 각 서브 트리 사이의 간격을 조사해서 좌표를 설정하기 때문에 노드들의 많은 이동이 있다. 일반 트리 도시 알고리즘은 후위 우선 순회에 의해 트리 내의 모든 노드들을 한번씩 방문하면서 트리의 좌표를 설정하기 때문에 노드수가 N개인 트리에서 시간 복잡도(*time complexity*)가 *O(N)*이다. 즉, 선형시간(linear time)에 수행하는 알고리즘이므로 다음과 같은 정리를 얻는다.

정리 3.3

일반 트리 도시 알고리즘에의한 도시는 O(N)안에 실행된다.(여기서 N은 트리의 노드 수이다.)



[그림-3. 일반 트리 도시]

4. 결론

다양한 응용분야를 갖는 트리를 도시하는 알고리즘은 이미 많이 존재하고 있다. 이들 알고리즘의 목표는 최소의 공간을 사용하여 트리의 미적 효과를 증대 시켜 이해도와 판독성을 얼마만큼 올리는가이다. 트리를 그릴 때에 필요한 공간을 최소화하는 데만 중점을 둔다면, 자칫 이해도를 떨어뜨릴 수 있다는 단점이 있다. 따라서 최소의 면적을 이용하여 미적 효과를 최대화 시킬 수 있는 알고리즘이 필요하다.

Tilford의 알고리즘은 트리를 도시하는데 필요한 면적을 최소로 한다. 그러나 왼쪽을 중심으로 도시하기 때문에 트리가 왼편으로 치우치며, 노드의 불필요한 이동이 많다는 단점을 가지고 있다. 본 논문에서는 Tilford 알고리즘의 단점을 개선하여 일반 트리 도시 알고리즘을 제시하였다. 그 결과 노드의 불필요한 이동횟수를 줄였으며 손쉽게 트리를 도시할 수 있다는 장점을 지닌다.

참고문헌

[1] J. S. Tilford, "Tree drawing algorithm", *technical Report, Department of Computer Science, Univ. of Illinois at Urbana, Rep. UIUC DCS-R-81-1055, Arizona State Univ.*, 1981.
[2] G. M Radack, "Tidy drawings of M-ary trees", *technical Report, Department of Computer Science, Case Western Reserve Univ., Cleveland, Ohio,* November 1988.
[3] C. Wetherell and A. Shannon, "Tidy drawings of trees", IEEE Trans, Software Eng, vol SE-5, pp 514-520, 1979.
[4] P. Crescenzi, G. Di Battista, and A. Piperno, "A Note on Optimal Area Algorithms for Upward Drawings of Binary Trees", Computational Geometry: Theory and Applications, vol. 2 pp 187-200, 1992.