Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Information Processing Letters 88 (2003) 225–229

Information Processing Letters

www.elsevier.com/locate/ipl

Salembier's Min-tree algorithm turned into breadth first search

Wim H. Hesselink

Department of Mathematics and Computing Science, Rijksuniversiteit Groningen P.O. Box 800, 9700 AV Groningen, The Netherlands

Received 22 May 2003; received in revised form 4 August 2003

Communicated by S. Albers

Keywords: Graph algorithms; Spanning tree; Connected component; Threshold set

1. A graph-theoretic problem

Consider a connected undirected graph (V, E) with a real-valued function f defined on the vertices. For real h, let the h-threshold set V_h be the set of vertices x with $f(x) \leqslant h$. The algorithm of Salembier e.a. [4] computes the connected components of all sets V_h in a single computation. Here, we present a simplification of the algorithm together with a proof of correctness.

Let us use the term *component* to indicate a connected component of some V_h. The components form a tree in the sense that V itself is the root and, for every pair of nondisjoint components, one of the pair contains the other. We determine the tree of components by giving a parent pointer to every node. In every component, a root node with maximal f-value is chosen. The parent pointer points from a nonroot node to the root node of its smallest component, from a root node to the root node of the unique smallest enclosing component, and from the root node of V to $\perp$. Following [4], such a pointer structure is called a Min-tree.

We refer to [2–4] for the relevance of Min-trees (or their duals, called Max-trees) for image processing. Our use of parent pointers to characterize components was inspired by Tarjan's union-find algorithm [5], used in a similar way in [1].

The algorithm of Salembier et al. [4] constructs a Min-tree by means of a recursive procedure that contains three nested loops with recursive calls in the innermost loop. We replace this recursive procedure by a single repetition, by turning the procedure inside-out upon a suggestion of J.E. Jonker.

The parent pointers are collected in a function $\mathtt{par}: V \to V \cup \{\perp\}$. By convention $f(\perp) = \infty$. Function $\mathtt{par}$ is called *ascending* iff

$$\text{(Asc)} \quad f(x) \leqslant f\big(\mathtt{par}[x]\big) \quad \text{for all } x \in V.$$

Given $\mathtt{par}$, a vertex x is called a *level root* iff $f(x) < f(\mathtt{par}[x])$. The set of descendants of a vertex x is defined by

$$D(x) = \big\{y \in V \mid \exists\, n \in \mathbb{N}\colon \mathtt{par}^n[y] = x\big\}.$$

Notice that $x \in D(x)$ since $\mathtt{par}^0[x] = x$.

We define a *Min-tree* to be an ascending function $\mathtt{par}$ such that every component is of the form $D(x)$ for some level root x. If $\mathtt{par}$ is a Min-tree, function

E-mail address: wim@cs.rug.nl (W.H. Hesselink).
URL: http://www.cs.rug.nl/~wim.

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2003.08.003

D represents a bijective correspondence between the level roots and the components:

Lemma 1. *Let* par *be a Min-tree.*

(a) *For every level root x, the set $D(x)$ is a component.*
(b) *If x and y are level roots with $D(x) = D(y)$, then $x = y$.*

Proof. (a) Level root x is contained in some component U of the threshold set $V_{f(x)}$. Since par is a Min-tree, $U = D(y)$ for some level root y. It follows that $y = \mathtt{par}^n[x]$ for some natural number n. If $n \geqslant 1$, then $f(x) < f(y)$ since x is a level root and par is ascending. Since $y \in U \subseteq V_{f(x)}$, it follows that $n = 0$, and hence $x = y$ and $D(x) = U$. (b) This is proved in the same way. □

2. The Min-tree constructed

We write $\mathsf{Nhb}(x)$ to denote the set of neighbors of a node x. The algorithm uses the following additional variables

W, T: **set of** Node;
wait: **array** $\mathbb{R}$ **of set of** Node;
root: **array** $\mathbb{R}$ **of** Node $\cup \{\bot\}$.

W stands for the set of nodes that have been reached by the algorithm. T holds level roots. The set wait$[h]$ holds neighbors of nodes of level h that have yet to be investigated. Variable root$[h]$ is used to hold a level root at level h. Initially, all sets are empty and all elements of par and root are $\bot$:

$W = T = \emptyset \wedge (\forall\, h \in \mathbb{R}\colon \mathtt{wait}[h] = \emptyset)$,
$(\forall\, x \in V\colon \mathtt{par}[x] = \bot) \ \wedge \ (\forall\, h \in \mathbb{R}\colon \mathtt{root}[h] = \bot)$.

Here, and henceforth, the symbol $\wedge$ stands for conjunction (logical and). The variable T can be eliminated from the algorithm, but will be used in the proof.

The algorithm starts at an arbitrary node xm at an arbitrary level lev.

Alg:

choose xm $\in V$;
lev $:= f(\mathtt{xm})$; $W := \{\mathtt{xm}\}$;
root[lev] := xm; wait[lev] := Nhb(xm);
while lev $< \infty$ **do**
  if wait[lev] $\neq \emptyset$ **then** Encounter
  else Upward **end**
end.

The commands Encounter and Upward are defined as follows.

Encounter:

remove some nd from wait[lev];
if nd $\notin W$ **then**
  $W := W \cup \{\mathtt{nd}\}$;
  if root$[f(\mathtt{nd})] = \bot$ **then** root$[f(\mathtt{nd})] :=$ nd;
  else par[nd] := root$[f(\mathtt{nd})]$ **end**;
  wait$[f(\mathtt{nd})] :=$ wait$[f(\mathtt{nd})] \cup$ Nhb(nd);
  if $f(\mathtt{nd}) <$ lev **then** lev $:= f(\mathtt{nd})$ **end**;
end.

The value of lev never increases in Encounter, but always increases in

Upward:

determine minimal $m >$ lev
  with root$[m] \neq \bot \vee m = \infty$;
par[root[lev]] := root$[m]$;
$T := T \cup \{$root[lev]$\}$;
root[lev] $:= \bot$;
lev $:= m$.

The symbol $\vee$, used in the first line of Upward, stands for disjunction (logical or).

Informally speaking, the algorithm traverses the graph and chooses in every new level the first node it encounters as a root. The other nodes of that level get a parent pointer to the root. Once a component is reached, the algorithm is restricted to neighbors of nodes of the component until the component is visited completely. At that point, the root of the smallest enclosing component has been chosen, and the root of the component gets a parent pointer to that root.

Writing $\#S$ for the cardinality (number of elements) of a set S, the time complexity is linear in $\#E + \#V$, provided the first line of Upward can be done in constant time. The complication here is inherited from [4].

The algorithm uses sets wait$[h]$ rather than lists or queues, but that is also allowed, since the test nd $\notin$

W guards against multiple treatment and sets can be emptied in arbitrary order.

In applications with large graphs, one may want to bound the sizes of the sets $\texttt{wait}[h]$. A solution is to apply double buffering by introducing an array of sets $\texttt{buf}$ and replacing the line

$$\texttt{wait}[f(\texttt{nd})] := \texttt{wait}[f(\texttt{nd})] \cup \mathsf{Nhb}(\texttt{nd});$$

by $\texttt{buf}[f(\texttt{nd})] := \texttt{buf}[f(\texttt{nd})] \cup \{\texttt{nd}\}$, and to expand the neighborhood of $\texttt{nd}$ only when $\texttt{wait}[\texttt{lev}]$ needs new elements. Each set $\texttt{buf}[h]$ only needs the size $\#\{x \mid f(x) = h\}$.

The algorithm of [4] only uses array $\texttt{buf}$, the sets $\texttt{wait}$ are hidden in the recursion stacks. For Min-trees, the algorithm of [4] initializes $\texttt{xm}$ such that the value of $f(\texttt{xm})$ is maximal. Our version is more nondeterministic in the choices of $\texttt{xm}$ and $\texttt{nd}$.

3. The proof of correctness

The proof of the algorithm consists of three parts. We first prove termination by means of a variant function and five invariants (Iq0..4). This is followed by the easy part of functional correctness: $\texttt{par}$ is ascending and all nodes are reached by the algorithm, with invariants (Asc) and (Jq0..6). The difficult part of the proof consists of invariants (Kq0..2), which express how the algorithm traverses the components of the graph.

To prove termination, we define the variant function $\mathsf{vf} = \mathsf{vf0} + \mathsf{vf1} + \mathsf{vf2}$ by

$$\begin{aligned} \mathsf{vf0} &= (\sum h \in \mathbb{R}\colon \#\texttt{wait}[h]),\\ \mathsf{vf1} &= (\sum x \in V \setminus W\colon \#\mathsf{Nhb}(x)),\\ \mathsf{vf2} &= \#(V \setminus T). \end{aligned}$$

Here, as above, if S is a set, $\#S$ stands for its cardinality.

The repetition terminates, since $\mathsf{vf} \geqslant 0$ and the loop body decreases vf. Indeed, under its precondition $\texttt{wait}[\texttt{lev}] \neq \emptyset$, command $\mathsf{Encounter}$ decreases $\mathsf{vf0} + \mathsf{vf1}$ and does not modify $\mathsf{vf2}$. Command Upward does not modify $\mathsf{vf0}$ and $\mathsf{vf1}$, and decreases $\mathsf{vf2}$ because of the invariant

(Iq0) $\texttt{root}[h] \notin T$.

Here, we universally quantify over the free variable h. Predicate (Iq0) holds initially since initially T is empty. It is preserved under modification of T because of the easily verified invariants that $f(\texttt{root}[h]) = h$ for every h with $\texttt{root}[h] \neq \bot$, and that $\texttt{root}[\texttt{lev}] \neq \bot$ whenever $\texttt{lev} < \infty$, as expressed in

(Iq1) $\texttt{root}[h] \neq \bot \;\Rightarrow\; f(\texttt{root}[h]) = h$,
(Iq2) $\texttt{lev} < \infty \;\Rightarrow\; \texttt{root}[\texttt{lev}] \neq \bot$.

Predicate (Iq0) is preserved under the assignments to $\texttt{root}$ because of

(Iq3) $T \subseteq W$.

Invariance of (Iq3) follows from (Iq2) and the obvious invariant

(Iq4) $\texttt{root}[h] \in W \cup \{\bot\}$.

This concludes the proof of (Iq0) and thus proves that the repetition terminates. The number of executions of the loop body is bounded by the initial value of vf which equals $\#E + \#V$.

Since $f(\bot) = \infty$ and initially $\texttt{par}[x] = \bot$, it is easy to verify that (Asc) is an invariant of the algorithm: function $\texttt{par}$ is invariantly ascending. It is easy to verify that $\texttt{root}$ satisfies the two invariants

(Jq0) $h < \texttt{lev} \;\Rightarrow\; \texttt{root}[h] = \bot$,
(Jq1) $\texttt{root}[h] = \bot \;\Rightarrow\; \texttt{wait}[h] = \emptyset$.

Notice how the last instructions of $\mathsf{Encounter}$ and Upward preserve (Jq0). In order to prove the postcondition $W = V$, we observe the invariants

(Jq2) $\texttt{xm} \in W$,
(Jq3) $x \in W \;\Rightarrow\; \mathsf{Nhb}(x) \subseteq W \cup \texttt{wait}[f(x)]$.

The algorithm terminates with $\texttt{lev} = \infty$. By (Jq0), (Jq1), and (Jq3), this yields the postcondition $\mathsf{Nhb}(x) \subseteq W$ for every $x \in W$. Since the graph is connected and W is nonempty by (Jq2), this implies the postcondition

(Post0) $W = V$.

With respect to par, we have the easy invariants

(Jq4) $x \notin W \;\Rightarrow\; \mathtt{par}[x] = \bot$,
(Jq5) $\mathtt{root}[h] \neq \bot \;\Rightarrow\; \mathtt{par}[\mathtt{root}[h]] = \bot$,
(Jq6) $x \in T \;\Rightarrow\; f(x) < f(\mathtt{par}[x])$.

We use (Iq4) to prove preservation of (Jq5) under Encounter. Notice that (Jq6) says that T consists of level roots.

The components of level h are the connected components of the subgraph with V_h as set of vertices and $E_h = E \cap (V_h \times V_h)$ as set of edges. We regard E_h as a symmetric binary relation on V and write E_h^* for its transitive closure $(E_h)^*$, which is an equivalence relation on V. The components at level h are the equivalence classes of E_h^* contained in V_h.

In order to relate the components to function D, we first relate relation E_h^* to root and wait. For arbitrary real numbers h, k, and n, we claim the invariants

(Kq0) $\mathtt{root}[k] \neq \bot \neq \mathtt{root}[n] \;\wedge\; \max(k, n) \leqslant h \;\Rightarrow\; (\mathtt{root}[k], \mathtt{root}[n]) \in E_h^*$,
(Kq1) $x \in \mathtt{wait}[k] \;\wedge\; \max(k, f(x)) \leqslant h \;\Rightarrow\; (\mathtt{root}[k], x) \in E_h^*$.

Before proving these invariants, we note that (Kq0) and (Kq1) imply that the assignments to wait, root and par in Encounter have the precondition

PreC: $\mathtt{root}[k] \neq \bot \;\wedge\; \max(k, f(\mathtt{nd})) \leqslant h \;\Rightarrow\; (\mathtt{root}[k], \mathtt{nd}) \in E_h^*$.

Indeed, since $\mathtt{root}[k] \neq \bot$, we have $\mathtt{lev} \leqslant k$ by (Jq0), so that (Kq0) and (Iq2) imply that E_h^* contains the pair $(\mathtt{root}[k], \mathtt{root}[\mathtt{lev}])$. Since nd is taken from $\mathtt{wait}[\mathtt{lev}]$, predicate (Kq1) implies that E_h^* contains $(\mathtt{root}[\mathtt{lev}], \mathtt{nd})$. By transitivity, this implies $(\mathtt{root}[k], \mathtt{nd}) \in E_h^*$, thus proving PreC.

Predicate (Kq0) is threatened by the assignment to $\mathtt{root}[f(\mathtt{nd})]$ in Encounter. By symmetry, we may assume that $n = f(\mathtt{nd})$ and $\mathtt{root}[n] := \mathtt{nd}$. Therefore, (Kq0) is preserved because of precondition PreC.

Predicate (Kq1) is threatened by the assignments to root and wait. If $\mathtt{root}[k] = \bot$, these assignments preserve (Kq1) because of (Jq1) and the observation that

$x \in \mathsf{Nhb}(\mathtt{nd}) \;\wedge\; \max(f(\mathtt{nd}), f(x)) \leqslant h \;\Rightarrow\; (\mathtt{nd}, x) \in E_h$.

With respect to the assignment to $\mathtt{wait}[k]$ when $\mathtt{root}[k] \neq \bot$, consider $\mathtt{nd} \in \mathtt{wait}[\mathtt{lev}]$ with $f(\mathtt{nd}) = k$ and $x \in \mathsf{Nhb}(\mathtt{nd})$ and $\max(k, f(x)) \leqslant h$. Now PreC yields $(\mathtt{root}[k], \mathtt{nd}) \in E_h^*$ and $(\mathtt{nd}, x) \in E_h$ by the definitions of E_h and Nhb. This implies $(\mathtt{root}[k], x) \in E_h^*$. This concludes the proofs of (Kq0) and (Kq1).

We turn to the sets $D(x)$ as defined in the introduction. $D(x)$ can only change when par is modified. In the algorithm, every assignment $\mathtt{par}[r] := s$ has the precondition $\mathtt{par}[r] = \bot$ because of (Jq4) and (Jq5). It follows that $D(x)$ can only grow: if $y \in D(x)$ holds, it remains valid.

We now consider a level h and a connected component U of the subgraph (V_h, E_h). The aim is to prove that the algorithm establishes $U = D(x)$ for some $x \in T$. For this purpose, we introduce a predicate (Kq2) which is a disjunction of three alternatives: Alt0 tends to hold initially; Alt2 is the goal of the argument; the intermediate predicate Alt1 asserts that lev is below the value h under consideration, and that the intersection $U \cap W$ is the union of the descendant sets $D(\mathtt{root}[k])$ where k ranges over the levels $k \leqslant h$ with $\mathtt{root}[k] \neq \bot$. For conciseness, we use the convention that $D(\bot)$ is empty. Using this, (Kq2) is defined by

(Kq2) $\mathsf{Alt0} \vee \mathsf{Alt1} \vee \mathsf{Alt2}$, where
Alt0: $U \cap W = \emptyset$,
Alt1: $\mathtt{lev} \leqslant h \wedge U \cap W = \bigcup_{k \leqslant h} D(\mathtt{root}[k])$,
Alt2: $(\exists\, x \in T\colon U = D(x))$.

Notice that the sets $D(\mathtt{root}[k])$ are disjoint because of (Jq5). Therefore, the union in Alt1 is a disjoint union.

We prove that (Kq2) is an invariant. We have $W = \{\mathtt{xm}\}$ initially. Therefore, if $\mathtt{xm} \notin U$, then Alt0 holds initially. Otherwise, Alt1 holds initially since, because of $\mathtt{xm} \in U$, the initial value of lev is $\leqslant h$ and $U \cap W = \{\mathtt{xm}\} = D(\mathtt{xm})$. This proves that (Kq2) holds initially.

Now assume that Alt0 holds and a loop body falsifies Alt0. This means that Encounter finds a node $x_0 \in U$, adds it to W, and establishes $\mathtt{lev} = f(x_0) \leqslant h$ and $U \cap W = \{x_0\}$. The precondition satisfies $x_0 \in$

wait[lev] and root[lev] $\in W$ by (Iq2) and (Iq4). We have $(\texttt{root}[\texttt{lev}], x_0) \notin E_h^*$ since $\texttt{root}[\texttt{lev}] \notin U$. By (Kq1), this implies the precondition $h < \texttt{lev}$ and hence $\texttt{root}[k] = \bot$ for all $k \leqslant h$ by (Jq0). We therefore have $\{x_0\} = \bigcup_{k \leqslant h} D(\texttt{root}[k])$ in the postcondition of this Encounter.

Next, assume that Alt1 holds and a node $\texttt{nd} \notin W$ is encountered. The precondition satisfies $\texttt{lev} \leqslant h$ and $\texttt{nd} \in \texttt{wait}[\text{lev}]$. Alt1 and (Iq2) imply $\texttt{root}[\texttt{lev}] \in U$. Using (Kq1) with $x := \texttt{nd}$, $k := \texttt{lev}$, one can prove that nd belongs to U if $f(\texttt{nd}) \leqslant h$. Conversely, $\texttt{nd} \in U$ implies $f(\texttt{nd}) \leqslant h$, since $U \subseteq V_h$. It follows that nd is added to the set $U \cap W$ if and only if $f(\texttt{nd}) \leqslant h$. By the assignment to par or root, node nd is added to $D(\texttt{root}[f(\texttt{nd})])$. This implies that it is added to $\bigcup_{k \leqslant h} D(\texttt{root}[k])$ if and only if $f(\texttt{nd}) \leqslant h$. This shows that Alt1 is preserved.

Next assume that Alt1 holds and Upward is executed with $x_1 = \texttt{root}[\texttt{lev}]$ and determined value m. If $m \leqslant h$, the value of root[lev] is set to $\bot$, but the assignment to par adds the elements of $D(x_1)$ to $D(\texttt{root}[m])$, so that Alt1 is preserved. So, assume that $m > h$. Then $\texttt{root}[k] = \bot$ for all k with $k < m \ \wedge \ k \neq \texttt{lev}$. Therefore Alt1 implies $U \cap W = D(x_1)$. By (Jq1), we have $\texttt{wait}[k] = \emptyset$ for all $k < m$. We show that this implies $U \subseteq W$. For every $x \in U \cap W$ and every y with $(x, y) \in E_h$, we have $y \in U$ and $\texttt{wait}[f(x)] = \emptyset$ and hence $y \in W$ by (Jq3), so that $y \in U \cap W$. Since $U \cap W$ is nonempty and U is a connected component of V_k, this implies $U \subseteq U \cap W$ and hence $U \subseteq W$ and $U = D(x_1)$. Since Upward adds x_1 to T, this establishes Alt2.

Once Alt2 holds, it remains valid, since (Iq0) and (Jq5) imply that the set $D(x)$ for $x \in T$ never changes. This concludes the proof that (Kq2) is an invariant.

In the postcondition of the algorithm, we have $\texttt{lev} = \infty$ and $W = V$ by (Post0). Therefore (Kq2) yields the postcondition

(Post1) $\quad (\exists\, x \in T\colon\ U = D(x))$.

Since par is always ascending and T consists of level roots by (Jq6), postcondition (Post1) implies

Theorem 2. *Algorithm* Alg *establishes that* par *is a Min-tree.*

Actually, the algorithm establishes somewhat more. As is easy to verify, T is the set of level roots and $\texttt{par}[x] \in T \cup \{\bot\}$ for all vertices x.

References

[1] W.H. Hesselink, A. Meijster, C. Bron, Concurrent determination of connected components, Sci. Comput. Programm. 41 (2001) 173–194.

[2] W.H. Hesselink, J.E. Jonker, A. Meijster, M.H.F. Wilkinson, Accumulating attributes over growing connected components, in preparation.

[3] A. Meijster, M.H.F. Wilkinson, A comparison of algorithms for connected set openings and closings, IEEE Trans. Pattern Analysis and Machine Intelligence 24 (2002) 484–494.

[4] P. Salembier, A. Oliveras, L. Garrido, Anti-extensive connected operators for image and sequence processing, IEEE Trans. Image Process. 7 (1998) 555–570.

[5] R.E. Tarjan, Efficiency of a good but not linear set union algorithm, J. ACM 22 (1975) 215–225.