Seminar on Algorithms and Data Structures: Multiple-Edge-Fault-Tolerant Approximate Shortest-Path Trees [1]

Philipp Rimle

April 14, 2018

1 Introduction

Given a graph with positively real-weighted undirected edges, common problems including finding a minimum-spanning-tree (MST), where the sum of the edges is to be minimized, or a shortest-path-tree (SPT), where we fix a source s and minimize the sum of the edge lengths from the source to each vertex. Figure 1 presents a small example with a MST and a SPT of a sample graph. One can imagine that finding an MST or an SPT of a graph is useful in any kind of network where we want to optimize the path-lengths between the vertices. A more concrete problem would be in communication networks, where we want to broadcast messages from a source to all or some receivers in an optimal way.

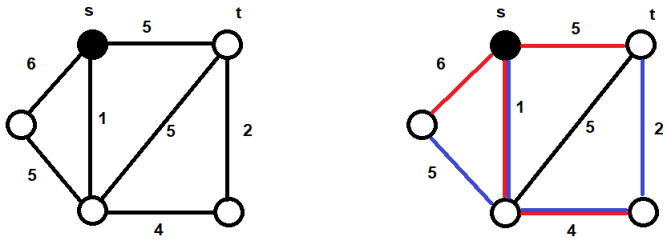


Figure 1: Example graph where the MST and the SPT differentiate. The MST of the sample graph is highlighted in blue and the SPT is highlighted in red. The total cost of the SPT is 16 whereas the cost of the MST is 11. However we can easily see that the MST is not a possible SPT by looking at the path s to t, which has a cost of 7 in the MST but has a shortest path of 5 in the graph

The problem with tree-based topologies is that they are highly sensitive to edge and vertex failures, which would disconnect the vertices from each other. We could therefore try to make an SPT resistant against an amount of failing edges or vertices by adding suitably selected edges from the underlying graph to it. Unfortunately for a graph G with n vertices and m edges, the amount of edges we would need to add, even for a single failing edge and $|V(G)| = O(n^2)$ vertices, would already be θ(m). Thus it makes sense to sparsify the structure

and then try to approximate the shortest paths from the source, bounding them by a stretch factor $\sigma > 1$.

In the underlying paper [1], they presented the method of building an f-EFT $(2|F|+1)$-ASPT H of a graph G with size $|E(H)| = O(fn)$, that is able to handle the failure of any set $F \subseteq E(G)$ of at most f edges. These terms are defined as follows.

f-EFT σ-ASPT Given a graph $G = (V(G), E(G), w)$ with m positively real-weighted undirected edges and n vertices together with a fixed source $s \in V(G)$. A spanning subgraph H of G is an f-edge-fault-tolerant (f-EFT) σ-approximate-shortest-path-tree (ASPT) if it satisfies the following condition: For each set of edges $F \subseteq E(G)$ of size at most f, all the distances from the source s in the subgraph $H - F = (V(G), E(H) \setminus F, w)$ are at most σ times longer than the corresponding distances in $G - F$.

The paper thus presents a sparse subgraph H with $|E(H)| = O(fn)$ in which the shortest paths from the source to any vertices after at most f edges failing, do not get stretched more than a factor of $\sigma = 2|F|+1$ compared to the original underlying graph G with the same set of failed edges. The paper focuses only on edge failures and not on how to protect a structure against vertex failures. In a second step the authors show how to transform such a structure into an efficient oracle that will allow to quickly switch to alternative paths in case some edge failures happen. More precisely they show how to construct a f-EFT $(2|F|+1)$-SSDO of size $O(min\{m, fn\}log^2 n)$ that has a query time for a post-failure distance from the source of $O(|F|^2 log^2 n)$ and is also able to report the corresponding path in the same time plus the path size.

f-EFT σ-SSDO A counterpart of the f-EFT SPT structures are f-edge-fault-tolerante (f-EFT) σ-stretched single-source distance oracles (SSDO), compact data structures that can quickly return σ-approximate distances/paths from the source following a set of failures. This allows us to quickly compute the alternative distance/paths on the structure after some failures.

2 Algorithm

We now look at the algorithm described in the paper which computes the subgraph H, which is an f-EFT $(2|F|+1)$-ASPT of G. And then we have a look why the algorithm is correct.

Algorithm 1 works as follows: For a graph G, a fixed source s and a given integer f, which is our upper bound for the amount of edges which can fail ($|F| <= f$), we first compute a shortest-path-tree T of G (Illustration in Figure 2). Let $d_x(u, u')$ denotes the distance between the vertices u and u' in any subgraph X of G. If $u = s$ we will simply write $d_X(u')$. The Algorithm then creates an auxiliary graph $G' = (V(G), E(G), w')$, where the weight of an edge $e = (u, v)$ in G' is 0 if e is also in T and otherwise is equal to the sum of the corresponding edge weight in G and the distances in T between the source s and the endpoints of e ($w'(e) = d_T(u) + w(e) + d_T(v)$). In the second step we iteratively compute $f + 1$ minimum-spanning-forest (MSF) $M_0, \ldots, M_f$ of G_i,

Algorithm 1: Algorithm for computing a f-EFT $(2|F|+1)$-ASPT of G described in the paper

$T \leftarrow$ compute an SPT of G
for $(u,v) \in E(G)$ **do**
  if $(u,v) \in E(T)$ **then**
    $w'(u,v) \leftarrow 0$
  else
    $w'(u,v) \leftarrow d_T(u) + w(u,v) + d_T(v)$
  end
end
$G' \leftarrow (V(G), E(G), w')$
$G_0 \leftarrow G'$
for $i = 0, \cdots, f$ **do**
  $M_i \leftarrow$ edges of an MSF of G_i (w.r.t. w')
  $G_{i+1} \leftarrow G_i \setminus M_i$
end
$H \leftarrow$ subgraph of G containing the edges in $\bigcup_{i=0}^{f} M_i$
return H

where $G_0 = G'$ and we remove the edges of the computed M_i in G_i which gives us G_{i+1}, before we compute the next MSF M_{i+1} in G_{i+1}.

A first observation we make before proving the algorithm is that the created graph H is indeed sparse in a sense that it only contains $|E(H)| = O(fn)$ edges. This can easily be seen in the algorithm since the resulting subgraph H only consist of $f+1$ many pairwise disjoint minimum spanning forests M_i.

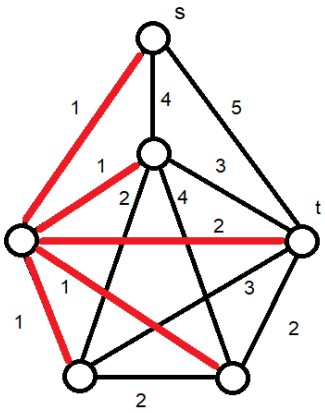
3 Stepwise Proof with Core Ideas

In order to proof the correctness of the algorithm and hence Theorem 5, we proceed stepwise by looking at some lemmas first, which can be combined in a later stage. Let us start with Lemma 1 which we will later use in Lemma 3.

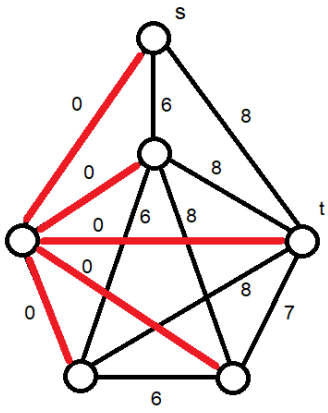
Lemma 1 *For every $F \subseteq E(G)$ with $|F| \leq f$, any MSF M of $H - F$ (w.r.t. w') is also an MSF of $G' - F$ (w.r.t. w').*

In H we have the disjoint MSF M_0 up to M_f which always contain the "lightest" edges in the remaining graph starting from G' and removing the respective MSF. Therefore for any given cut-set (set of edges whose removal increases the connected components of a graph) C of G', we see that H contains the min $\{|C|, f+1\}$ lightest edges of C. Looking at an edge e of the MSF M' of $G' - F$ and the cut-set C' containing this edge e together with all the edges $e' \in E(G')$ which form a cycle with e in $M' \cup \{e'\}$, one can see that e is within the lightest $f+1$ edges in C' since e is the lightest edge of $C' \setminus F$. Therefore if the amount of failing edges is smaller or equal than f, e is an edge in $H - F$, and since it is the lightest edge in $C' \cap E(H-F)$ it also belongs to the MSF of $H - F$.

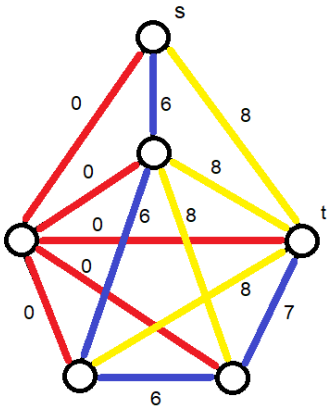
In the next step, we want to give an upper bound to a path from the source s to some fixed vertex t in our surviving graph $H - F$. Let $\pi' = \pi_M(t)$ be



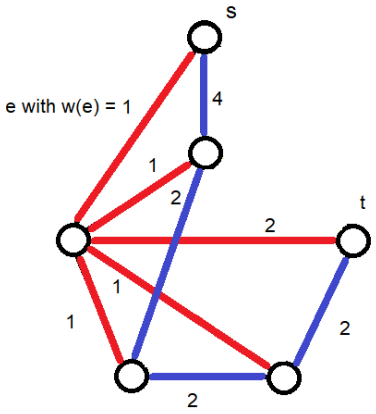
(a) SPT T (highlighted in red) of G rooted in the source s



(b) Auxiliary graph G' of G with the weights $w'(u,v) = d_T(u)+w(u,v)+d_T(v)$



(c) The (disjoint) MSF M_0 (red), M_1 (blue) and M_2 (yellow) highlighted in G'



(d) Subgraph H of G, calculated by Algorithm 1 with $f = 1$

Figure 2: Illustration of Algorithm 1 on a sample graph G. In (a) we see the SPT T of G rooted in the source s. In the first part of the Algorithm 1 we compute the auxiliary graph G' with the corresponding weights w' shown in (b). In the second part of the algorithm we compute the pairwise disjoint MSF M_i in G' shown in (c). Subfigure (d) shows a resulting subgraph H of G for $f = 1$ and therefore includes the MSF M_0 and M_1. First note that H is indeed sparse and fulfils the criteria of an 1-EFT $(2 * 1 + 1)$-ASPT. Consider for example the path s to t in (d) in a scenario where the edge e fails. Hence the distance from s to t in $G \setminus \{e\}$ is 5, whereas the distance in $H \setminus \{e\}$ is $7 \leq 15 = 3 \cdot 5 = (2 \cdot f + 1) \cdot 5 = \sigma \cdot d_G(s,t)$

the unique path from s to some fixed vertex t in the MSF M of the surviving graph $H - F$. First note that the path π' is traversing each tree of the forest $T - F$ at most once since the edges in $E(T)$ have weight 0 in H according to the construction described in Algorithm 1. We can think of the trees of the forest $T - F$ as rooted in the vertex which is closest to s in the original SPT T. Further let N' be the set of new edges in π'. We say an edge is new if its endpoints belong to two different trees in the remaining forest $T - F$. Note that T was an SPT of G and by removing edges of the set F, it can get disconnected (into no more than $|F| + 1$ connected components). Figure 3 shows a path π' in $H - F$, taking some new edges between the trees of the forest $T - F$.

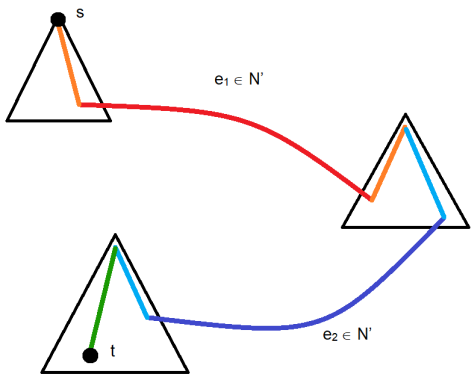


Figure 3: Illustration of a path π' in $H - F$ going through the trees of the forest $T - F$ from s to t. We can see that the distance of the path π' (w.r.t w) is covered by the sum of the weights (w.r.t w') of the new edges e_1 and e_2 plus the remaining distance of the path from the root of the last tree to t, which is upper bounded by the distance $d_G(t)$.

Lemma 2 $d_{H-F}(t) \leq w(\pi') \leq \sum_{e \in N'} w'(e) + d_G(t)$.

When we look closer at the weight w' of an edge, we see that by definition it is the weight w of the edge in G plus the distance from the source to the two endpoints in the SPT T. It therefore includes the edge weight w(e) and the weight of the path in the two trees (part of the MSF $T - F$) of the endpoints of e. We see an illustration of this composition in Figure 3, where the weight (w.r.t. w) of the red and orange path is already paid in w' of the new edge $e_1 \in N'$ and the weight of the light and dark blue path is already paid off in the new edge $e_2 \in N'$. As a result one can see that the total weight of π' (w.r.t. w) is indeed covered by the sum of the weights w' of all new edges $e \in N'$ plus the weight of the path in the last tree to the vertex t. This last path is again upper bounded by just taking the distance from s to t in the original graph G. Note that $d_G(t)$ looks like an over-approximation. However, it turns out that a more restrictive approach would lead to the same result in the worst-case scenario.

Let $\pi = \pi_{G-F}(t)$ be the shortest path from s to some fixed vertex t in the surviving graph $G - F$ (We assume this path exists, otherwise the $d_{G-F}(t) = +\infty$ which implies $d_{H-F}(t) = +\infty$) and let N similarly to N' be the set of new

edges in π. Now we want to show that the weights of the new edges in π' are upper bounded by the weight of some new edge of the path π.

Lemma 3 *For each $e' \in N'$, we have $w'(e') \leq max_{e \in N} w'(e)$.*

Let us look at a situation where the paths π and π' do not traverse the same set of trees in the forest $T - F$. Let Z be the set of trees that are traversed by the path π and let $e' \in N'$ be a new edge of the path π', which is not between two trees in Z (see Figure 4). Further let N^* be the set of new edges in the path π, which are between the tree where π' is leaving the set Z and the tree where π' is entering the set Z again. Note that if we add an edge from N^* to the MSF M of the surviving graph $H - F$, it would form a cycle containing both e' and an edge in N^*, say e^*. We showed in Lemma 1 that M is also an MSF in $G' - F$ and therefore we have that $w'(e') \leq w'(e^*)$. As this holds for some $e^* \in N^*$, it trivially also holds for the new edge with the highest weight (w.r.t w') and we can write $w'(e') \leq w'(e^*) \leq max_{e \in N} w'(e)$.

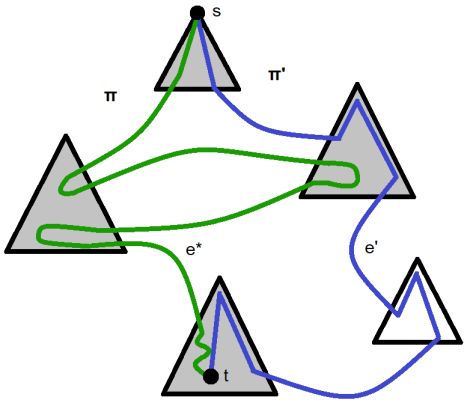


Figure 4: Illustration of the forest $T - F$ with the shortest path π between s and t in $G - F$ and the unique path π' in M between the same vertices. Grey trees are in the set Z which are traversed by the path π.

Let us now see how the weights w' of the new edges of π relate to the distances (w.r.t. w) in the surviving graph $G - F$.

Lemma 4 *For $e \in N$, $w'(e) \leq 2d_{G-F}(t)$.*

Let $e = (u, v)$ with $d_{G-F}(u) \leq d_{G-F}(v)$. Since e lies on the shortest path $\pi = \pi_{G-F}(s, t)$, we can write:

$w'(e) =_{def} \underline{d_T(u) + w(e)} + \underline{d_T(v)} \leq \underline{d_{G-F}(v)} + \underline{d_T(v)} \leq 2d_{G-F}(v) \leq 2d_{G-F}(t)$.

Theorem 5 *The graph H returned by Algorithm 1 is an f-EFT $(2|F|+1)$-ASPT of G. Moreover, Algorithm 1 requires $O(fm\alpha(m, n))$ time and $O(m)$ space.*

We first observe that $\pi' = \pi_M(s, t)$ contains at most $|F|$ new edges. Remember that all the edges in $T - F$ have weight 0 in H and the remaining edges

have a positive weight. That means that $E(T-F) \subseteq E(M)$ and as $T-F$ has no more than $|F|+1$ connected components, we have at most $|F|$ other edges which are not in $E(T-F)$ belonging to M. We can now write:

$$\begin{aligned} d_{H-F}(t) &\leq_{L2} w(\pi') \leq_{L2} \sum_{e \in N'} w'(e) + d_G(t) \\ &\leq |F| max_{e \in N'} w'(e) + d_G(t) \\ &\leq_{L3} |F| max_{e \in N} w'(e) + d_G(t) \\ &\leq_{L4} 2|F| d_{G-F}(t) + d_{G-F}(t) = (2|F|+1) d_{G-F}(t) \end{aligned} \tag{1}$$

After we have seen an upper bound for the stretching factor σ, let us look at the running time of Algorithm 1 to create a subgraph H. It can easily be seen that the running time is given by computing the f+1 MSF $M_0, \ldots, M_f$. Each MSF can be computed with the Chazelle's algorithm [2] which runs in $O(m\alpha(m,n))$ where α is the inverse of the Ackermann's function.

4 A Corresponding Oracle

We now want to find out how we can build an oracle which gives us an $(2|F|+1)$-approximated path from s to t in $G-F$. Notice that the alternative path is exactly π' which we have seen above. For the oracle we first need to compute a SPT T of G and a f-EFT $(2|F|+1)$-ASPT H of G. Then the oracle consists of the following three parts:

- the tree T and all the distances $d_T(v) = d_G(v)$ from s to any vertex $v \in V(G)$
- an MSF sensitivity oracle Q of H w.r.t. the weights w'
- an oracle to answer lowest common ancestor (LCA) queries between two vertices in T

Lets first have a look what a sensitivity oracle Q is and how we can construct it. Basically we want to report all the edges of T that leave the MST due to updates, along with all the edges in T' that enter the MST in their place. Exploiting the fact that if few updates have to be handled (as we have, since we need it only for the failing edges which is limited), the changes in the MST are small. The highlevel idea is now to build a structure that maintains a set of connected subtrees of T at different granularity. Such a hierarchical clustering of the vertices T can be seen in Figure 5 (a). Each cluster has a level $l(C) \in \{0, \cdots, L\}$ and we can group the clusters of a level i to a set C_i. The clustering is constructed that the following properties hold:

- Clusters of each level i are a partition of the vertices of T, i.e. they are pairwise disjoint and $\bigcup_{C \in C_i} C = V(G)$
- The vertices in each cluster induce a connected component of T
- Clusters of level 0 are singletons, i.e. they contain a single vertex of T
- There is only one cluster of level L (and it coincides with V(T))

- Each cluster of level $i \geq 1$ is the union of at least 2 and at most Δ clusters of level $i-1$, where Δ is the maximum degree of a vertex in T.

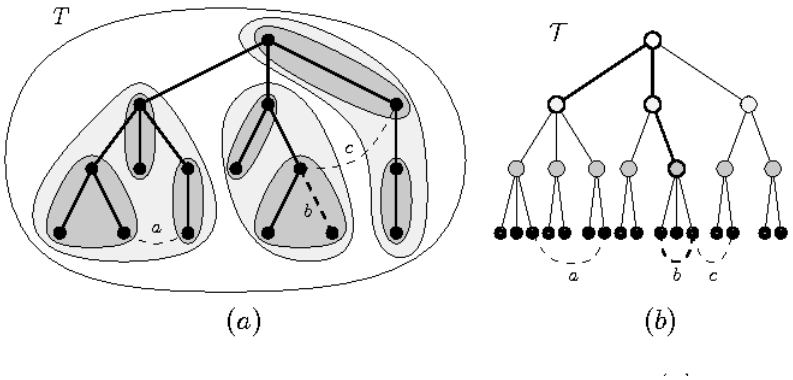


Figure 5: Figure from the paper [1], where we see a hierarchical clustering of the vertices of T. (a) shows the different clusters at different levels in T and (b) shows the hierarchical tree of the clusters

We can see that a cluster of level i contains at least 2^i vertices (because of the properties described above on how we create the clusters) and the maximum level is $L \leq log\ n$. In Figure 5 (b) we see the hierarchy of the clusters. The height is L and the root is the unique cluster C_L which consist of all the vertices of the MST T. We now maintain for each pair of clusters C, C' with $C \neq C'$ an ordered (by weights, ascending) set $E(C, C')$ containing all the edges of $E(G)$ with one endpoint in C and the other in C'. If we have a set F of failing edges, we process all the edges of F which are in the same cluster, by splitting the cluster on an auxiliary graph in its subclusters of level $i-1$. We will keep track of the different clusters and the resulting forest of the hierarchical tree $\mathcal{T}$. What we get in the end is that all the edges in F have their endpoints in different clusters in our auxiliary graph. For all pairs C, C' of vertices in our auxiliary graph we select an edge $E(C, C')$ which is not in F, if any, and add the edge to our auxiliary graph which now connects the two clusters C, C'. We can now compute an MST $\widetilde{T}$ in our auxiliary graph and thereby find the edges which are in $E(\widetilde{T})$ but not in $E(T)$, i.e. $E(\widetilde{T}) \setminus E(T)$.

A nice side note is that so far the complexity of the oracle depends on the maximum degree Δ of a vertex in T. This follows of how we construct the auxiliary graph. We can however avoid having a high degree by reducing the maximum degree of a vertex to at most 3. Figure 6 of the paper shows these technique where for each vertices with more than 2 children, we simply construct a binary tree with edgeweight 0 between the vertex ant its neighbours.

Coming back to our original f-EFT $(2|F|+1) - SSDO$ we can now think of how to answer path queries or distance queries which return a $(2|F|+1)$-approximate path between s and t or the corresponding distance. Remember that we want to return the path $\pi' = \pi_M(s, t)$. For that we first query the MSF oracle Q (informally described above) to get the set of new edges of the unique path from s to t in the updated MSF. Let $e'_i = (v_{i-1}, u_i)$ for $0 < i < h$ be the new edge from v_{i-1} to u_i, let r_i be the LCA (least common ancestor) between u_i and v_i, and let r_h be the LCA between u_h and t. For $\pi'_i = \pi_T(u_i, r_i) \circ \pi_T(r_i, v_i)$,

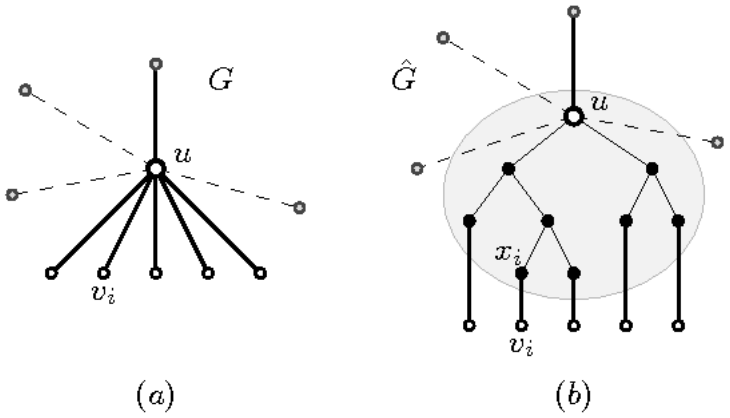


Figure 6: Figure from the paper [1] where we see how to reduce the degree of vertices in T

we can construct the approximated path as follow:

$$\pi' = \pi_T(s, v_0) \circ e'_1 \circ \pi'_1 \circ e'_2 \circ \pi'_2 \circ \cdots \circ \pi'_{h-1} \circ e'_h \circ \pi_T(u_h, r_h) \circ \pi_T(r_h, t)$$

If we want to report the distance, we just replace each subpath in the above equation with the corresponding distance.

5 Conclusion

The paper provided a way of computing a sparse f-EFT $(2|F|+1)$-ASPT of size $O(fn)$. In this way one can make an SPT of a graph more resistant against edge failures in a sense of bounding the shortest paths by a stretch factor of $(2|F|+1)$. It continued on a topic with already substantial research in similar questions and improved previous related constructions of f-EFT ASPT's designed for unweighted graphs. It furthers shows a way how a theoretical construction of an efficient f-EFT SSDO can be built. A question which this paper does not address is how the new structure performs against vertex failures and how to build structures that are resilient against both edge and vertex failures.

References

[1] Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Fault-tolerant approximate shortest-path trees. *CoRR*, abs/1407.0637, 2014.

[2] Bernard Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. *J. ACM*, 47(6):1028–1047, November 2000.