Definitions: Define a problematic vertex as a vertex with degree greater than two. A vulnerable problematic vertex is a problematic vertex with at most one child that is not in a leaf path. An leaf subpath is a subpath which has one endpoint being a problematic vertex and one endpoint having degree one. An internal subpath is a subpath whose endpoints are both problematic vertices, or the root. We define the problem size Σ as the number of vertices a feasibility test needs to check,

$$\Sigma = PV + l_1 + 2l_2 + 3l_4 + 4l_8 + 5l_{16} + \dots,$$

where PV is the number of unresolved problematic vertices and l_{2^i} is the number of cleaned and glued paths whose length is between $2^{i-1} + 1$ and 2^i. The algorithm will maintain two selection lists, L_1 and L_2. The first selection list is for paths, while second selection list is for handling problematic vertices whose leaf paths have been resolved.

Assumptions: For each problematic vertex v with children vertices $c_1, \cdots, c_k$, where $k > 2$, we create a complete binary tree rooted at v, whose bottom layer has $2^j \geq c_k >$

2^{j-1} vertices, including $c_1, c_2, \cdots, c_k$. All other vertices in the tree have vertex weight zero. Thus, we now have a binary tree while at most tripling the number of total vertices. Consequently, we can transform any given tree to a binary tree.

Algorithm: Initialize the algorithm. Run Phase 1 l times, where l is a constant which is later defined. Then run Phase 2 a single time. While the weight inserted into L_1 at the end of Phase 2 does not exceed the weight remaining in L_1 and the runtime on problematic vertices and resolved internal subpaths is at most half of the runtime, run Phase 1 l times and Phase 2 a single time. When the runtime on problematic vertices and resolved internal subpaths is at least half of the runtime, repeatedly run Phase 2 until there is no weight in L_2. When the weight insert into L_1 at the end of Phase 2 exceeds the remaining weight in L_1, we continue the process, but use a different form of analysis.

Initialization: Given a tree T with n vertices initially, take the edge-path-partition of T and remove all problematic vertices. Take each subpath P_i that remains and extend the length to t_i, the smallest possible power of two while labeling from the root, by adding in vertices with vertex weight zero, as necessary. For each subpath P_i, create an associated matrix, and insert its largest and smallest values into selection set L_1 with weight $\lceil \frac{4n^4}{t_i} \rceil$. For each subpath, repeatedly split the subpath, and insert the largest and smallest nonzero values of the associated matrices into L_1 with twice the weight of the previous weight. For each problematic vertex, insert paths of length of 2^i into D with weight $\lceil \frac{4n^4}{2^i} \rceil$ while t_i is less than two times the maximum distance of a vertex from the root. Finally, for each internal subpath, extend the length to 2^i, the smallest possible power of two longer than its length, and insert into D with weight $\lceil \frac{4n^4}{2^j} \rceil$, where 2_j is the smallest possible power of two longer than i..

Phase 1: Select the weighted and unweighted medians from L_1, test for feasibility, and adjust λ_1 and λ_2 accordingly. During the feasibility testing, glue and clean adjacent subpaths which have been resolved. If any leaf path has been completely resolved, the leaf path can be represented by a single vertex.

Phase 2: For any vulnerable problematic vertex with a resolved leaf path, insert into L_2 the weight of that vertex plus the accumulated remaining weight from the resolved leaf path, for the max-min problem. Select the median from L_2, perform the feasibility test, and adjust λ_1 and λ_2 accordingly. In the max-min problem, if any vulnerable problematic vertices are resolved in this phase, we transfer the corresponding subpaths into L_1 from D with the appropriate weight. Furthermore, if any internal path becomes a leaf path following the feasibility test, we remove the corresponding weight from D.

Theorem 1 *When the weight inserted by L_2 into L_1 is less than the weight in L_1, running the selection on L_1 five times reduces the weight remaining in L_1 by a factor of 1/6.*

Proof By Lemma 3.2 of the path partition paper, each iteration decreases the weight remaining in L_1 by a factor of at least 1/6. Suppose at the end of iteration i, the weight remaining in L_1 is W. Since $(5/6)^4 < 1/2$, then selecting the median of L_1 and running the feasibility test five times leaves L_1 with at most $(1/2)(5/6)W$ weight. Since the weight inserted by L_2 is less than the remaining weight in L_1, the weight in L_1 is at most $(5/6)W$ at the end of the iteration. ■

Theorem 2 *When the weight inserted by L_2 into L_1 is less than the weight in L_1, the i-th iteration of the algorithm will spend amortized time $O(i(5/6)^{i/5}n)$ on the unresolved internal subpaths and the leaf paths.*

Proof By similar reasoning to Lemma 3.2, each 1×1 submatrix quartered repeatedly from a matrix representing a path of length 2^j has weight $n^4/2^{4j-5}$. By iteration i, where

i satisfies $(4/3) * 4n^5 * (5/6)^i = n^2/2^{5j-5}$ or $2^{5j} = 6 * (6/5)^i$. However, at most $1/2^{5k}$ of the subpaths of length 2^{j-k} can be unresolved and at most $(1/2^{5j-2})$ of the subpaths of length 1 can be unresolved. By Lemma 3.1, each subpath can be searched in amortized time proportional to its length, so the time to search the leaf subpaths and unresolved internal subpaths on iteration i is at most

$$(j/2^j)n(1 + 1/2^5 + 1/2^{10} + \cdots) = O((j/2^j)n).$$

Since $2^j = 6^{1/5}(6/5)^{i/5}$, then $j = (1/5)\log 6 + (i/5)\log(6/5)$. ■

Corollary 3 *When the runtime on problematic vertices and resolved internal subpaths is at most half of the runtime, the i-th iteration of the algorithm will spend amortized time $O(i(5/6)^{i/5}n)$ on the unresolved internal subpaths and the leaf paths.*

Theorem 4 *When the weight inserted by L_2 into L_1 is less than the weight in L_1 and no leaf paths are completely resolved, it is not possible for the runtime on problematic* *vertices and resolved internal subpaths to be at most half of the runtime.*

Proof Suppose on iteration i, where i satisfies $(4/3) * 4n^5 * (5/6)^i = n^2/2^{5j-5}$, there exists l such that there are N resolved internal subpaths of length 2^{j-l}, where $N > (1/2^{5k})\left(\frac{n}{2^{l-k}}\right)$. Since there are no resolved leaf paths, there must be N unresolved paths of average length $j - l$ for the runtime on problematic vertices and resolved internal subpaths to be at most half of the runtime. However, since $j - l < 2^{j-l}$, the previous theorem guarantees there cannot be N unresolved paths of average length $j - l$. ■

Claim 5 *When the weight inserted by L_2 into L_1 is at least the weight in L_1, the amortized time for resolving and releasing m problematic vertices is better than the amortized time for resolving and releasing 1 problematic vertex m times.*

Claim 6 *When a single problematic vertex is resolved and released and the weight inserted by L_2 into L_1 is at least the weight in L_1, the runtime for feasibility testing is $O(c + \log^2 n)$, where c is the number of remaining problematic vertices.*

Claim 7 *The amortized time for resolving all resolved leaf paths in Phase 2 is $O(n)$.*