Algorithm C below is a simplified algorithm for determining strict $L_\infty$ isotonic regression of an arbitrary dag. It uses a simple array and a single sort instead of the dynamic priority queue used in Algorithm B. The time constants should be quite small.

If the dag has a transitive closure of $m'$ pairs then the time is $\Theta(m' \log m')$, the same as Algorithm B (this does not count the time to determine the transitive closure, which is part of the input). In terms of the original dag with $n$ vertices this is at most $\Theta(n^2 \log n)$. A more detailed time analysis gives $\Theta(m' + m^* \log m^*)$, where $m^*$ is the number of violating pairs of vertices, i.e., vertices $u, v$ such that $u \prec v$ and $f(u) > f(v)$. The only component taking $\Theta(m^* \log m^*)$ time is the sort, with all other lines combined taking only $\Theta(m')$. Thus, for a fixed dag, the more isotonic the data is, the faster the algorithm.
input: weighted data \((f, w)\), lists of successors and predecessors for each vertex
output: strict \(L_\infty\) isotonic regression function \(S\)
violators: array of \((\text{mean	extunderscore error}, u, v)\) for violating pairs \(u \prec v, f(u) > f(v)\)
lowbd\((v)\), upbd\((v)\): lower and upper bounds on \(S(v)\)

\[
\text{numviolate}=0
\]
for every vertex \(v\)
\[
\text{lowbd}(v) = -\infty; \quad \text{upbd}(v) = +\infty; \quad S(v) = \text{undefined}
\]
for every successor \(s\) of \(v\)
\[
\text{if } f(v) > f(s) \text{ then } \text{violators(numviolate)} = (\text{mean	extunderscore err}(v,s), \ v, \ s); \quad \text{numviolate}++
\]

sort violators by \(\text{mean	extunderscore err}\)

for \(i=0\) to \(\text{numviolate}-1\)
\[
(\text{mean	extunderscore err}, \text{pred}, \text{suc})=\text{violators}(i)
\]
if \((S(\text{pred}) \text{ defined}) \lor (S(\text{suc}) \text{ defined})\) then cycle
\[
\text{wmean} = \text{mean}(\text{pred},\text{suc})
\]
if \(\text{wmean} \geq \text{upbd}(\text{pred})\) then \(\{f(\text{pred}) \geq \text{upbd}(\text{pred}), \text{no later mean is } < \text{upbd}(\text{pred})\}\)
\[
S(\text{pred}) = \text{upbd}(\text{pred})
\]
if \(\text{wmean} \leq \text{lowbd}(\text{suc})\) then \(\{f(\text{suc}) \leq \text{lowbd}(\text{suc}), \text{no later mean is } > \text{lowbd}(\text{suc})\}\)
\[
S(\text{suc}) = \text{lowbd}(\text{suc})
\]
if \((S(\text{pred}) \text{ undefined}) \land (S(\text{suc}) \text{ undefined})\) then \(\{\text{low}(\text{suc}) \leq \text{wmean} \leq \text{high}(\text{pred})\}\)
\[
S(\text{pred}) = S(\text{suc}) = \text{wmean}
\]
if \(S(\text{pred})\) defined then
for every successor \(s\) of \(\text{pred}\)
\[
\text{lowbd}(s) = \max\{\text{lowbd}(s), S(\text{pred})\}
\]
if \(S(\text{suc})\) defined then
for every predecessor \(p\) of \(\text{suc}\)
\[
\text{upbd}(p) = \min\{\text{upbd}(p), S(\text{suc})\}
\]
end for \(i\)

for every vertex \(v\)
\[
\text{if } S(\text{v}) \text{ undefined then}
\]
\[
\text{if } f(\text{v}) \geq \text{upbd}(\text{v}) \text{ then } S(\text{v})=\text{upbd}(\text{v})
\]
\[
\text{else if } f(\text{v}) \leq \text{lowbd}(\text{v}) \text{ then } S(\text{v})=\text{lowbd}(\text{v})
\]
\[
\text{else } S(\text{v})=f(\text{v})
\]

**Algorithm C**: Computing \(S=\text{Strict}(f,w)\) using transitive closure