Reductions to Sink-of-DAG are Compressed

Nathan Acheampong Jake Gameroff

Supervised by Professor Robert Robere

Our reduction moves between two total search problems, which we record here before turning to the proof.

- **Sink-of-DAG**, denoted SoD_N: Given an $N \times N$ grid where each node either has a self-loop or points to a node one level down. Return $(1, 1)$ if it has a self-loop, or any other node with a self-loop and a predecessor. This is equivalent to the problem of finding a sink in a general DAG.
- **Iteration**, denoted Iter_N: This is the same as SoD but on an $N^2 \times 1$ grid and with the additional property that each node can point to any node under it (as opposed to just one level down).

Lemma 1. *If* $\text{A}_n \leq_d \text{SoD}_N$ *then* $\text{A}_n \leq_d \text{Iter}_{N^2}$.

Proof. From the SoD_N instance we have vertices $V = \{(i, j) : 1 \leq i, j \leq N\}$. We define an ordering on these vertices by $(a, b) < (c, d)$ if and only if

$$(a = c \text{ and } b < d) \quad \text{or} \quad (a < c).$$

Following this ordering, we can write $V = \{v_i : 1 \leq i \leq N^2\}$. Then any edge $e = (v_i, v_j)$ satisfies $i < j$, and so this gives us an Iter_{N^2} instance. Notice also that for an input x, o solves SoD_N if and only if it solves Iter_{N^2}. Since this is, effectively, just a re-labelling, we get the reduction for free. ■

Lemma 2. *If* $\text{A}_n \leq_d \text{Iter}_N$ *then* $\text{A}_n \leq_d \text{SoD}_N$.

Proof. We take an instance of Iter_N on a graph $G = (V, E)$, with vertices $v_1, v_2, \ldots, v_N$ and self-loops allowed (denoting a nil successor). We will define an auxiliary graph H on the vertices $V_H = [N] \times [N]$ which will help us encode the Iter instance G as a sink-of-dag instance.

Each vertex $v_i \in V$ from the Iter_N instance will have some successor s_{v_i}. We will use column $i + 1$ in H to "remember" that the edge $(v_i, s_{v_i}) \in E$. This leaves all of the columns allocated, except for the first. This column will store the solutions (i.e. sinks), and edges (v_i, v_{i+1}) if present.

First, if there is a loop at the vertex v_i in G, then we just put a loop at the vertex $(i, 1)$ in H. Otherwise, we have $(v_i, v_j) \in E$ for some $j > i$. Again, we will use column $i + 1$ to capture this relationship. We first add an edge from $(i, 1)$ to $(i + 1, i + 1)$. Then, we add an edge from $(i + 1, i + 1)$ to $(i + 2, i + 1)$, and so on, all the way until we get to $(j - 1, i + 1)$. Formally, for each $i + 1 \leq k \leq j - 2$, we add an edge from $(k, i + 1)$ to $(k + 1, i + 1)$. We then add one final edge from $(j - 1, i + 1)$ to $(j, 1)$. If any other vertex is edgeless after building H, just give it a self-loop.

Hence, under this construction, there is a path P from $(i,1)$ to $(j,1)$ in H if and only if $(v_i, v_j) \in E$. We additionally see that v is a solution to $\textsc{Iter}_N$ if and only if v is a solution in $\textsc{SoD}_N$. We thus again get the reduction for free: based on the d bits read from A_n, each sink-of-dag vertex just declares what its neighbour is according to the construction of H from G. ■

Lemma 3. *If* $\mathrm{A}_n \leq_d \textsc{Iter}_N$ *then* $\mathrm{A}_n \leq_{O(d)} \textsc{Iter}_{n^{O(d)}}$.

Proof. Consider the forward maps (f_{ij}) given by the reduction $\mathrm{A}_n \leq_d \textsc{Iter}_N$. Note that without loss of generality we may assume that every path in each f_{ij} has length d. Indeed, if there is a path of length $k < d$, we can just add in $d - k$ more dummy queries from the remaining unqueried variables and produce the same output. We will call two decision tree paths "identical" if they (i) query the same set of input bits (possibly in different orders) and (ii) for every queried variable x_i, both paths have the same edge-label (i.e. they both ask if $x_i = b$ for a bit $b \in \{0,1\}$).

Let P be the set of all decision tree paths in (f_{ij}). Consider any pair of identical paths in P, in the trees f_u, f_v with outputs o_u, o_v respectively. Suppose without loss of generality that $o_u \leq o_v$, according to the vertex-ordering defined in Lemma 1. If we have equality, do nothing (we will come back to this); otherwise, we rewire the output of f_u to be o_v. After this process, let $\mathcal{O}$ be the set of all possible outputs over all trees (f_{ij}). What kind of bound can we establish on $|\mathcal{O}|$?

Note that there are at most $\binom{n}{d}2^d$ pairwise non-identical paths in (f_{ij}). Hence, because of the rewiring step, it follows that $|\mathcal{O}| \leq \binom{n}{d}2^d = n^{O(d)}$. Now, for any $o \notin \mathcal{O}$, it is clear that we can safely delete the tree f_o. Furthermore, for every $o \in \mathcal{O}$, define the set S_o of all trees f_{ij} which output o. We will arbitrarily delete all but one tree from each S_o. Let us then re-label the trees to get $f_1, f_2, \ldots, f_k$. Notice that $k \leq n^{O(d)}$.

We claim that we have enough tools to reduce A_n to $\textsc{Iter}_{n^{O(d)}}$ in depth d. The trees (f_i) transform the input to A_n into an input for $\textsc{Iter}_{n^{O(d)}}$. For any solution o to $\textsc{Iter}_{n^{O(d)}}$, we already have a tree T_o which maps o back to a solution to A_n. Fix an input x and suppose $(f(x), o) \in \textsc{Iter}_{n^{O(d)}}$. Then by construction $(f(x), o) \in \textsc{Iter}_N$, where this f is the original set of maps given by the assumed reduction, and so $(x, T_o(x)) \in \mathrm{A}_n$. It then follows that $\mathrm{A}_n \leq_d \textsc{Iter}_{n^{O(d)}}$. ■

Theorem 1. *If* $\mathrm{A}_n \leq_d \textsc{SoD}_N$ *then* $\mathrm{A}_n \leq_{O(d)} \textsc{SoD}_{n^{O(d)}}$.

Proof. Since $\mathrm{A}_n \leq_d \textsc{SoD}_N$, Lemma 1 implies that $\mathrm{A}_n \leq_d \textsc{Iter}_{N^2}$. Then, Lemma 3 implies that $\mathrm{A}_n \leq_{O(d)} \textsc{Iter}_{n^{O(d)}}$. We then invoke Lemma 2 to conclude that $\mathrm{A}_n \leq_{O(d)} \textsc{SoD}_{n^{O(d)}}$. ■