1. COMPUTATIONAL EFFICIENCY

One computational hurdle of our active learning algorithm is to efficiently calculate the closure set $C_{(x,y)}$ given a complete G-oracle W_H. In particular, among all the formula in Theorem 1, we found the main bottleneck is to efficiently calculate $O_{(x,y)}$ whose worst time complexity is $O(|H|^3)$ (followed by Prop. 1), while others can be done in $O(|H|^2)$.

PROPOSITION 1. *Following the notations in Theorem 1, given $S_{(a,b)} \cup T_{(a,b)}$, the worst time complexity to calculate $O_{(a,b)}$ is $O(|H|^3)$.*

PROOF. If $(c,d) \in S_{(a,b)}$, $c \in D_b^{G\cap H} \cup \{b\}$. then $c \notin A_a^{G\cap H} \cup \{a\}$, thus $D_c^{G\cap(H\cup N_{(a,b)})} = D_c^{G\cap H}$ (by Definition of $N_{(a,b)}$). It holds that $D_c^{G\cap(H\cup N_{(a,b)})} \cup \{c\} \subseteq D_b^{G\cap H} \cup \{b\}$. Therefore, one has $N''_{(c,d)} \subseteq N''_{(b,d)} \subseteq O_{(a,b)}$ if $(c,d) \in S_{(a,b)}$. Likewise, $N''_{(c,d)} \subseteq N''_{(c,a)} \subseteq O_{(a,b)}$ if $(c,d) \in T_{(a,b)}$. One has

$$O_{(a,b)} = \left[\bigcup_{(c,d)\in S_{(a,b)}} N''_{(b,d)} \right] \cup \left[\bigcup_{(c,d)\in T_{(a,b)}} N''_{(c,a)} \right],$$

whose time complexity is $O(|H|^3)$. □

Using the following Prop. 2, one can show that there exists a pruning rule that cuts a major proportion of redundant set operations in calculating $O_{(a,b)}$.

PROPOSITION 2. *Following the notations in Theorem 1, we have $N''_{(c',d')} \subseteq N''_{(c,d)}$ if $(c',d') \in N''_{(c,d)}$.*

PROOF. Let $(c',d') \in N''_{(c,d)}$. If $d' \neq d$, $(d',d) \in G \cap (H \cup N_{(a,b)})$. Then, $A_{d'}^{G\cap(H\cup N_{(a,b)})} \subseteq A_d^{G\cap(H\cup N_{(a,b)})}$ (AAA). Thus,

$$A_{d'}^{G\cap(H\cup N_{(a,b)})} \cup \{d'\} \subseteq A_d^{G\cap(H\cup N_{(a,b)})} \cup \{d\}.$$

Likewise,

$$D_{c'}^{G\cap(H\cup N_{(a,b)})} \cup \{c'\} \in D_c^{G\cap(H\cup N_{(a,b)})} \cup \{c\}.$$

By the definition of $N''_{(c',d')}$ and $N''_{(c,d)}$, one has $N''_{(c',d')} \subseteq N''_{(c,d)}$. □

We also conduct empirical studies to examine the growth rate of calculating $C_{(x,y)}$. In practice, we find the empirical growth rate is closer to linear rate, which means the worst time complexity bound presented here is very conservative.

2. BOUNDS ON THE NUMBER OF QUERIES

Note that any strict order G can be described as a directed acyclic graph (DAG). We show the lower and upper bounds on the number of queries that are needed to learn a consistent classifier for G.

THEOREM 2. *Given a strict order G of V, let A be a consistent learner that makes m queries to the oracle before termination, then*

$$|\underline{G}| \leq m \leq |\overline{G}| \tag{1}$$

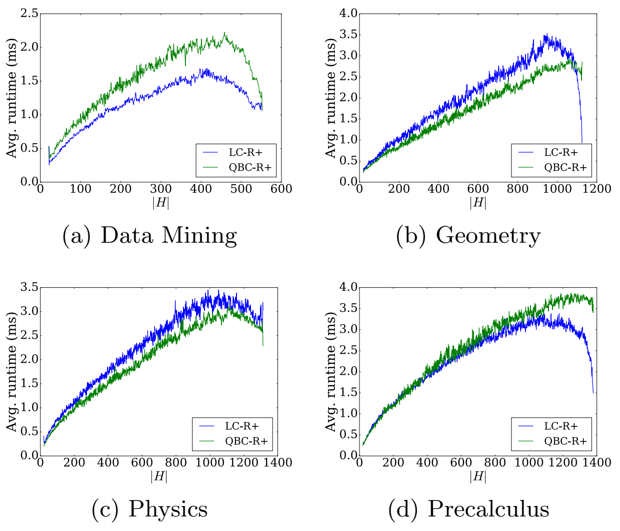
where $\underline{G}$ is the transitive reduction [1] of G and $\overline{G}$ is the closure (Def. 4) of G.

The above lower and upper bounds are tight in the sense that there exist DAGs G_1 and G_2, such that $|\underline{G_1}| = |G_1|$ and $|\overline{G_2}| = |G_2|$. With the power of the active learning paradigm, we want to empirically show that the number of queries needed is much smaller.

3. EFFICIENCY STUDY

The proposed reasoning module is designed to be plugged into any algorithm that needs reasoning of strict orders. Thus besides verifying effectiveness, it is also important to investigate its efficiency. We conduct empirical studies on the runtime of the reasoning module.

Figure 2 shows the relation between the average runtime for calculating the new closure using Theorem 1 and the size of the current labeled closure $|H|$. Results for both LBC-R+ and QBC-R+ are presented. We can see that as $|H|$ keeps increasing during the pool-based active learning process, the average runtime of calculating $C_{(x,y)}$ increases almost linearly and even decreases a little at the end. Although the worst case time complexity for calculating Theorem 1 is $O(|H|^3)$ (for $O_{(a,b)}$) and $O(|H|^2)$ (for others), the runtime required is directly related to the number of ascendants and descendants of elements in V, which is usually different for the four strict order datasets used. If the few ascendants and descendants effectively control the size of calculations as we have observed, the runtime will be short regardless of a large $|H|$. This might explain why the growth of calculating $C_{(x,y)}$ is near linear. We also empirically evaluate the effects of using Prop. 2 on the efficiency and include the results in the supplemental material.



(a) Data Mining (b) Geometry

(c) Physics (d) Precalculus

Figure 2: The average runtime for calculating the new closure using Theorem 1 v.s. the size of current labeled closure $|H|$.

3.1 Effectiveness of Proposition 2

We also empirically evaluate the effects of using Proposition 2 on the efficiency. Specifically, we measure the total runtime of $O_{(a,b)}$ calculation in Theorem 1 for a full round of active learning (until $\mathcal{D}_u = \emptyset$) with and without applying the pruning rule induced by Proposition 2. The results are shown in Table 1 where the numbers are the average runtime over all 300 different rounds of active learning for each dataset. We can see that the pruning can lead to a speedup of 40-55% in the LC-R+ experiments and a speedup of 49-55% in the QBC-R+ experiments, which shows that Proposition 2 is helpful for higher efficiency.

Domain	**LC-R+**		**QBC-R+**	
	w/o pruning	**w/ pruning**	**w/o pruning**	**w/ pruning**
Data mining	5.5	3.3 (-40%)	5.5	2.6 (-53%)
Geometry	85.5	39.2 (-54%)	69.6	31.4 (-55%)
Physics	111.0	58.8 (-47%)	117.1	60.2 (-49%)
Precalculus	134.3	59.9 (-55%)	167.4	74.8 (-55%)

Table 1: The effect of Proposition 2 on the total runtime (s) of $O_{(a,b)}$ calculation in Theorem 1 for a full round of active learning (until $\mathcal{D}_u = \emptyset$).

4. PROOFS

PROPOSITION 3. *For any $H \subseteq V \times V$, the closure of H subject to a strict order G is unique.*

PROOF. For any two supersets $H_1 \neq H_2$ of H whose oracles W_{H_1}, W_{H_2} are complete, $W_{H_1 \cap H_2}$, on a smaller set $H_1 \cap H_2$, is also complete (by definition). □

PROPOSITION 4. *Let G be a strict order of V. For a complete G-oracle W_H, $H \cap G$ is also a strict order of V.*

PROOF. For any $(a,b),(b,c) \in H \cap G$, because W_H is complete, $(a,c) \in H \cap G$ (by Definition 4 (i)). For any $(a,b) \in H \cap G$, $(b,c) \in H \cap G^c \subseteq (H \cap G)^c$ (by Definition 4 (iv)). Therefore, $H \cap G$ is also a strict order of V (by Definition 1). □

4.1 Proof of Theorem 1

4.1.1 Well-definiteness

It is trivial that if W_H is complete, $G \cap (H \cup N_{(x,y)})$ is also a strict order. Therefore, $N''_{(c,d)}$ is well defined, so is $O_{(x,y)}$.

4.1.2 Necessity

One can easily verify that if $(x,y) \in G \cap H^c$, both $N_{(x,y)} \subseteq \overline{H'}$ (Definition 3 (i)), $R_{(x,y)} \subseteq \overline{H'}$ (Definition 3 (iv)), and $S_{(x,y)} \cup T_{(x,y)} \cup O_{(x,y)} \subseteq \overline{H'}$ (Definition 3 (ii),(iii)) from the definition of closure, and likewise if $(x,y) \notin G$, $N'_{(x,y)} \subseteq \overline{H'}$ (Definition 3 (ii), (iii)). In another word, $C_{(x,y)}(H) \subseteq \overline{H'}$. Also, see Fig. 1 for the explanation of each necessary condition mentioned.

4.1.3 Sufficiency

One can see $A^G_{a'} \subseteq A^G_a$ if $a' \in A^G_a$ and $D^G_{a'} \subseteq D^G_a$ if $a' \in D^G_a$. That is, an ancestor of ancestor is also an ancestor (briefly, AAA), and a descendant of descendant is also a descendant (briefly, DDD).

Now we proceed to prove $C_{(x,y)}$ is complete using contradiction which finalizes the proof of our result $C_{(x,y)} = \overline{H'}$. If $C_{(x,y)}$ is not complete, by definition, one of the four conditions in Definition 3 must fail.

If Definition 3 (i) fails, there must exist a, b, c such that $(a,b) \in C_{(x,y)} \cap G$, $(b,c) \in C_{(x,y)} \cap G$, while $(a,c) \notin C_{(x,y)}$. In this case, if both (a,b) and (b,c) are from H, because H is complete, $(a,c) \in H \cap G$ contradicts the assumption. Hence at least one of (a,b) and (b,c) is not included in H. By the definition of $C_{(x,y)}$, if one pair belongs to $G \cap H^c$, it must come from $N_{(x,y)}$. Therefore, it implies $(x,y) \in G \cap H^c$.

$\underline{\text{Cases 1: If } (a,b) \in N_{(x,y)} \text{ and } (b,c) \in N_{(x,y)}}$, $(y,b) \in G$ and $(b,x) \in G$. That however implies $(y,x) \in G$, contradicting G's definition as a strict order (See Definition 1 (ii)).

$\underline{\text{Cases 2: If } (a,b) \in N_{(x,y)} \text{ and } (b,c) \in H}$, $a \in A^{G\cap H}_x$ and $c \in D^{G\cap H}_y$ (by DDD). It implies $(a,c) \in N_{(x,y)} \subseteq C_{(x,y)}$.

$\underline{\text{Cases 3: If } (a,b) \in H \text{ and } (b,c) \in N_{(x,y)}}$, $a \in A^{G\cap H}_x$ (by AAA) and $c \in D^{G\cap H}_y$. It implies $(a,c) \in N_{(x,y)} \subseteq C_{(x,y)}$.

In summary, Definition 3 (i) holds for $C_{(x,y)}$.

If Definition 3 (ii) fails, there must exist a, b, c such that $(a,b) \in C_{(x,y)} \cap G = G \cap (H \cup N_{(x,y)})$, $(a,c) \in C_{(x,y)} \cap G^c$, while $(b,c) \notin C_{(x,y)}$. In this case, if both (a,b) and (a,c) are from H, because H is complete, $(b,c) \in H \cap G^c$ contradicts the assumption. Hence *at least one of (a,b) and (a,c) is not included in H*. We divide the statement into the following cases to discuss:

$\underline{\text{Cases 1: If } (x,y) \in G, (a,b) \in N_{(x,y)}, \text{ and } (a,c) \in H \cap G^c}$, $(a,x) \in G \cap H$ and $(y,b) \in G \cap H$. Because $(a,x) \in H \cap G$, and $(a,c) \in H \cap G^c$, $(x,c) \in H \cap G^c$. Because $\{(x,y),(y,b)\} \subseteq G \cap (H \cup N_{(x,y)})$, $(x,b) \in G \cap (H \cup N_{(x,y)})$. Therefore, $(b,c) \in N''_{(x,c)} \subseteq C_{(x,y)}$.

$\underline{\text{Cases 2: If } (x,y) \in G \text{ and } (a,c) \in R_{(x,y)}}$, $(c,a) \in N_{(x,y)}$, thus $(c,b) \in N_{(x,y)}$. It implies $(b,c) \in R_{(x,y)} \subseteq C_{(x,y)}$.

$\underline{\text{Cases 3: If } (x,y) \in G \text{ and } (a,c) \in S_{(x,y)}}$, $(y,a) \in G \cap H$ and $\exists (d,c) \in G^c \cap H$ such that $(d,x) \in G \cap H$. Thus, $(x,c) \in G^c \cap H \Rightarrow (y,c) \in S_{(x,y)} \Rightarrow (b,c) \in N''_{(y,c)} \subseteq C_{(x,y)}$.

$\underline{\text{Cases 4: If } (x,y) \in G \text{ and } (a,c) \in T_{(x,y)}}$, $(c,x) \in G \cap H$ and $\exists (a,d) \in G^c \cap H$ such that $(y,d) \in G \cap H$. Thus, $(a,y) \in G^c \cap H \Rightarrow (a,x) \in T_{(x,y)} \Rightarrow (a,b) \notin N_{(x,y)}$. Because $(a,b) \in G \cap (H \cup N_{(x,y)})$, one has $(a,b) \in G \cap H \Rightarrow (b,x) \in T_{(x,y)} \Rightarrow (b,c) \in N''_{(b,x)} \subseteq C_{(x,y)}$.

$\underline{\text{Cases 5: If } (x,y) \in G \text{ and } (a,c) \in O_{(x,y)}}$, there exists $(d,e) \in S_{(a,b)} \cup T_{(a,b)}$ such that $(a,c) \in N''_{(d,e)}$. Therefore, $\{(a,b),(d,a),(c,e)\} \subseteq G \cap (H \cup N_{(x,y)})$. Hence, $(b,c) \in N''_{(d,e)} \subseteq C_{(x,y)}$.

$\underline{\text{Cases 6: If } (x,y) \in G^c}$, we have $(a,b) \in G \cap H$ and $(a,c) \in N'_{(x,y)}$. Thus $(x,b) \in G \cap H \Rightarrow (b,c) \in N'_{(x,y)} \subseteq C_{(x,y)}$.

In summary, all six cases above contradict the assumption $(b,c) \notin C_{(x,y)}$. Thus Definition 3 (ii) holds for $C_{(x,y)}$. Given we have verified the condition of Definition 3 (ii), one can also prove that Definition 3 (iii) holds in a similar way, because their statements as well as definitions of $S_{(x,y)}$ and $T_{(x,y)}$ are symmetric.

One can easily see that Definition 3 (iv) holds for $C_{(x,y)}$, because $G \cap (H \cup N_{(x,y)})$ is also a strict order and $R_{(x,y)} \subseteq C_{(x,y)}$. □

4.2 Proof of Theorem 2

We first introduce the notion of *transitive reduction* before we proceed:

Definition 6 (Transitive Reduction [1]). *Let G be a directed acyclic graph. We say $\underline{G}$ is a transitive reduction of G if:*

(i) There is a directed path from vertex u to vertex v in $\underline{G}$ iff there is a directed path from u to v in G, and

(ii) There is no graph with fewer arcs than $\underline{G}$ satisfying (i).

For directed acyclic graph G, Aho et al. [1] have shown that the transitive reduction is unique and is a subgraph of G. Let G be a simple directed acyclic graph (DAG). In compliance with Def. 4, we use $\overline{G}$ to denote the transitive closure of G. Define $S(G)$ as the set of graphs such that every graph in $S(G)$ has the same transitive closure as G, i.e.,

$$S(G) \coloneqq \{G' \mid \overline{G'} = \overline{G}\}$$

Aho et al. [1] have shown that $S(G)$ is closed under intersection and union. Further more, for DAG G, the following relationship holds:

$$\underline{G} = \bigcap_{G' \in S(G)} G' \subseteq G \subseteq \bigcup_{G' \in S(G)} G' = \overline{G}$$

Next, we give the proof of Theorem 2 below.

Proof. With the fact that negative labels from the query oracle cannot help to induce positive labels in the graph, we can bound the number of queries, m, needed to learn a classifier:

$$m \geq |\underline{G}|$$

The proof is by simple contradiction based on the definition of the transitive reduction of G and the fact that A is a consistent learner. On the other hand, there is a learning algorithm A that simply remembers all the queries with positive labels and predict all the other inputs as negative. For this algorithm A, it suffices for A to make $|\overline{G}|$ queries. □

5. EXPERIMENT ENVIRONMENT

All experiments are conducted on an Ubuntu 14.04 server with 256GB RAM and 32 Intel Xeon E5-2630 v3 @ 2.40GHz processors. Active learning query strategies are implemented in Python2.7. Code and data will be publicly available.

6. REFERENCES

[1] A. V. Aho, M. R. Garey, and J. D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972.