Appendix

RPCD ALGORITHM

We present RpCD algorithm in Alg. 1.

Algorithm 1 RpCD [Lee and Honavar, 2016a]

Input: $\mathcal{S}$ relational schema, h hop threshold

initialize $\boldsymbol{D}$ with candidate RDs up to h hops.
initialize an undirected graph $\mathcal{M}'$ with undirected $\boldsymbol{D}$.
$\ell \leftarrow 0$
repeat
  for $(P.Y, \mathcal{V}_X)$ **s.t.** $P.Y - \mathcal{V}_X \in \mathcal{M}'$ **do**
    for every $\boldsymbol{S} \subseteq ne(\mathcal{V}_X; \mathcal{M}') \setminus \{P.Y\}$ **s.t.** $|\boldsymbol{S}| = \ell$ **do**
      if $P.Y \perp\!\!\!\perp \mathcal{V}_X \mid \boldsymbol{S}$ **then**
        remove $\{P.Y - \mathcal{V}_X, \tilde{P}.X - \mathcal{V}_Y\}$ from $\mathcal{M}'$.
        break
  $\ell \leftarrow \ell + 1$
until $|ne(\mathcal{V}_X; \mathcal{M}')| - 1 < \ell$ for every $X \in \boldsymbol{A}$

initialize $\mathcal{U}$ with CUTs from $\mathcal{M}'$.
$\mathcal{N} \leftarrow \emptyset, \mathcal{H} \leftarrow \langle \boldsymbol{A}, \{X - Y \mid P.Y - \mathcal{V}_X \in \mathcal{M}'\}\rangle$
for every $\langle \mathcal{V}_X, \boldsymbol{P}.Y, R.Z\rangle \in \mathcal{U}$ **do**
  if $\langle X, Y, Z\rangle \in \mathcal{N}$ **or** $\{X, Z\} \cap ne(Y; \mathcal{H}) = \emptyset$ **or** $\{X, Z\} \cap ch(Y; \mathcal{H}) \neq \emptyset$ **then**
    continue
  if exists $\boldsymbol{S} \subseteq adj(\mathcal{V}_X; \mathcal{M}')$ **s.t.** $R.Z \perp\!\!\!\perp \mathcal{V}_X \mid \boldsymbol{S}$ **then**
    if $\boldsymbol{S} \cap \boldsymbol{P}.Y = \emptyset$ **then** orient $X \rightarrow Y \leftarrow Z$ in $\mathcal{H}$
    else if $X = Z$ **then** orient $Y \rightarrow X$ in $\mathcal{H}$
    else add $\langle X, Y, Z\rangle$ to $\mathcal{N}$
  orient edges in $\mathcal{H}$ with sound rules with $\mathcal{N}$.

$completes\,(\mathcal{H}, \mathcal{N})$
return $\bigcup_{P.Y - \mathcal{V}_X \in \mathcal{M}'} \begin{cases} P.Y \rightarrow \mathcal{V}_X & Y \rightarrow X \in \mathcal{H} \\ P.Y - \mathcal{V}_X & Y - X \in \mathcal{H} \end{cases}$

RELATIONAL DATA

We randomly generated 300 relational schemas of 3 (50%), 4 (25%), and 5 (25%) entity classes with specified probabilities. Two to five relationship classes are randomly generated to connect a pair (i.e., binary relationship) of entity classes or a triple with 75% and 25% probability, respectively. Cardinalities are selected uniformly. One to three attribute classes are generated for each entity class, and zero or one attribute class is generated for each relationship class. Finally, created relational schemas that do not satisfy following rules are excluded: (i) all item classes are connected and (ii) the total number of attribute classes are less than or equal to 8.

RCMs are also generated randomly. Given a relational schema $\mathcal{S}$, max hop length h is selected uniformly between 2 to 4. The number of dependencies is determined by $\lfloor \frac{3|\mathbf{A}|}{2} \rfloor$ and uniformly selected among *all* relational dependencies within the given h. We limit the maximum number of parents of a canonical relational variable by 3. We reject generated RCMs if there exists an isolated attribute class that does not involve any relational dependency. Further, if the CPRCM (a maximally-oriented RCM representing the Markov equivalence class of a RCM) of the generated RCM has no directed dependencies, that is, the orientation of relational dependencies is impossible in theory. We adopt a linear model with additive Gaussian noise using average aggregators:

$$i.X \leftarrow \sum_{P.Y \in pa(\mathcal{V}_X; \mathcal{M})} \frac{\beta_{P.Y, \mathcal{V}_X}}{|P.Y|_i^\sigma|} \left(\sum_{j.Y \in P.Y|_i^\sigma} j.Y \right) + \epsilon$$

where $\beta_{P.Y,\mathcal{V}_X} = 1 + |\gamma|$ where $\gamma \sim \mathcal{N}\left(0, 0.1^2\right)$ for every $P.Y \in pa\left(\mathcal{V}_X; \mathcal{M}\right)$ for every $X \in \mathbf{A}$. $\epsilon \sim \mathcal{N}\left(0, 0.1^2\right)$. The set of parameters will likely yield a relational data less hostile for our learning algorithm given that $\beta \geq 1$ and the variance of noise is relatively small. This fulfills our intention to assess the behavior of learning algorithm across different settings. If we wanted to exploit the fact that the generated RCMs are based on an average aggregator, we could incorporate this into the choice of kernel so that R-convolution kernel is not necessary but a simple RBF kernel on averaged values is sufficient.

Random relational skeletons are generated with a user-specified *base* size n. Given n, the number of relationships (i.e., relationship instances or relationship items) for each relationship class is the twice of the base size if the cardinality is 'many' for every its participating entity class and the same as base size, otherwise. The number of entities per entity class can be computed as $\lfloor 1.2^k \cdot n \rfloor$ where k is the number of related relationship classes with all-'one' cardinalities. For each RCM, we generate 4 relational skeletons corresponding to base size from 200 to 500, increased by 100.

ROBUSTIFICATION of RPCD VS. NAIVE RPCD

For the robust RpCD, we adopted features mentioned in the main paper: order-independence for Phase-I, and split-RBO, pair-RBO, non-RBO tests, and weak dependence detection for Phase-II. Aggregation-based additional tests are applied to both phases. Separating sets are sought from the smallest size of conditionals to the largest. If a separating set is found with size k, the algorithm checks the existence of other separating sets of the same size, which we call 'minimal separating sets'. Then, the orientation of relational dependencies is based on a majority vote for the orientation of each pair of attribute classes. At the end of the algorithm, different orientations are combined to yield a partially-oriented RCM (PRCM) which maximally satisfies obtained test results. If there are multiple candidate PRCMs matching the same number of test

Aggregation	Order Ind.	Base size	precision	recall
True	False	200	97.70%	63.71%
		300	97.61%	70.93%
		400	98.42%	76.21%
		500	98.81%	78.74%
	True	200	98.64%	60.12%
		300	98.99%	69.27%
		400	99.00%	74.52%
		500	99.04%	77.32%
False	False	200	98.02%	60.39%
		300	98.02%	68.40%
		400	98.32%	73.77%
		500	98.82%	76.29%
	True	200	98.76%	56.45%
		300	98.94%	66.23%
		400	98.86%	71.83%
		500	99.06%	75.03%

Table 5: Performance based on micro-average of Phase-I.

Base Size	Aggregation	Order Ind.	TP	FP
200	False	True	4.843	0.060
		False	5.143	0.090
	True	True	5.103	0.067
		False	5.350	0.130
500	False	True	6.373	0.057
		False	6.493	0.093
	True	True	6.523	0.063
		False	6.633	0.090

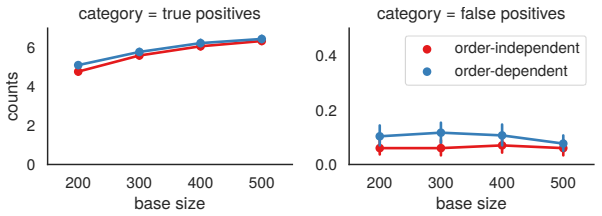
Table 6: Performance of Phase-I with average number of true positives (TP) and of false positives (FP)/ FPs are reduced to about two thirds by adopting order-independence.

results, then we choose a PRCM, which has the most common orientations with other competitors.

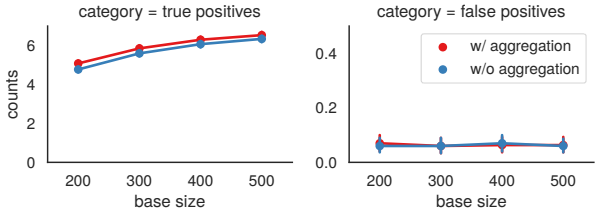
PHASE-I

We first report the performance of Robust RpCD for Phase-I. Micro-averaged precision and recall for undirected dependencies are reported (see Tab. 5). As the size of data increases, more accurate RCMs are discovered since RCI tests can better catch genuine dependencies. We observe relatively high precision in general even with a small-sized relational data, which implies that the main problem of the structure learning is false negatives due to weak dependencies.

Order-independence Fig. 4a depicts plots of performance with and without order-independence — the average number of true and false positives without additional aggregation-based tests. First, order-dependence can yield



(a) Performance on order-independence without aggregation



(b) Effect of aggregation with order-independence

Figure 4: Phase-I

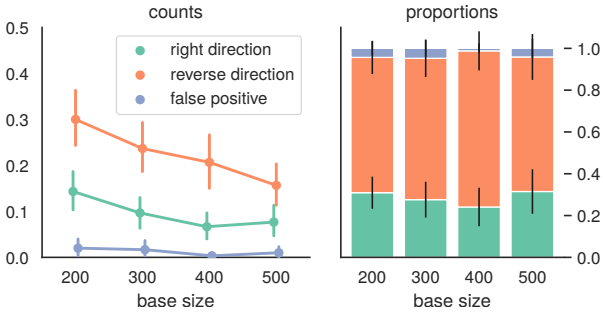


Figure 5: RCI query saved by aggregation-based tests

a higher number of both true and false positives. We can observe that order-independence reduces the number of false positives (see Tab. 6).

Aggregation In Fig. 4b, aggregation-based CI tests yield higher true positives without increasing false positives much. Since the non-aggregated test and its corresponding aggregated test are correlated, doubling the test does not significantly increase the false positive rate.

We explored which types of RCI queries are 'saved' by aggregated tests, i.e., $(U \perp\!\!\!\perp V \mid \mathbf{W}) \wedge (f(U) \not\perp\!\!\!\perp V \mid \mathbf{W})$ such that U is adjacent to V at the end of Phase I. We report three cases: i) false positive, $U \notin adj(V; \mathcal{M})$; ii) right direction, $U \in pa(V; \mathcal{M})$; and iii) reverse direction, $U \in ch(V; \mathcal{M})$. We expected that the aggregation-based test is particularly useful when $U \in ch(V; \mathcal{M})$ since V affects each of item attribute in U 'individually'. Then, averaging values might help reducing noises. In Fig. 5, we illustrate the average number of saved dependencies in the three categories and their proportions. Note that, an

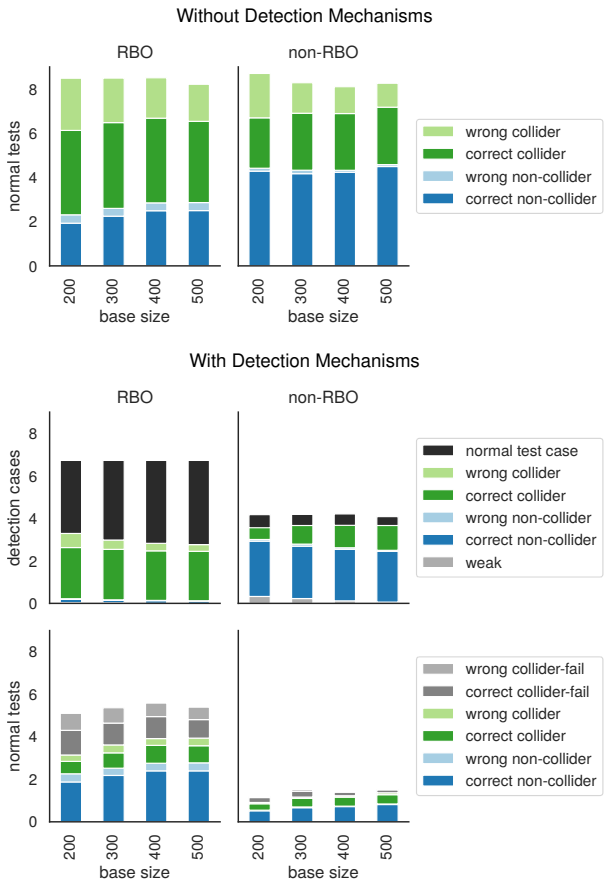


Figure 6: Effect of detection mechanism

adjacency $P.X - \mathcal{V}_Y$, which is also $\tilde{P}.Y - \mathcal{V}_X$, can be counted twice. We can first observe that the total number of saved relational dependencies decreases as data size increases since the original (i.e., non-aggregation-based) test will catch weak dependencies better. RCI tests in a reverse direction, e.g., $U \in ch\,(V; \mathcal{M})$, are mostly saved by aggregation. The use of aggregation will become more useful as the relationships in a relational skeleton become more complicated.

PHASE-II

We first overview how each feature affects the performance of orientation in terms of precision and recall assuming perfect Phase-I, which allows us to judge better how different features work. More specifically, 'correctly directed' relational dependencies lie in the intersection of oriented relational dependencies through Phase-II and true relational dependencies. Then, precision and recall are the proportion of correctly directed relational dependencies among directed relational dependencies through Phase-II, and among directed relational dependencies in the corresponding CPRCM, respectively.

Size	Precision	Recall	F-measure
200	65.8	61.0	63.3
300	74.2	67.8	70.8
400	71.9	66.6	69.1
500	75.3	69.7	72.4

Table 7: Orientation performance of a naive approach with CUT-based RCI tests.

Agg.	Size	Detection	Prec.	Recall	F
False	200	False	79.0	64.5	71.0
	300		84.7	68.7	75.8
	400		88.1	72.8	79.7
	500		87.8	73.6	80.1
False	200	True	88.6	69.4	77.8
	300		92.4	73.0	81.5
	400		94.2	76.2	84.2
	500		93.6	75.9	83.8
True	200	True	88.3	70.1	78.2
	300		91.5	73.6	81.6
	400		93.8	76.6	84.3
	500		93.5	75.4	83.5

Table 8: Orientation performance with our proposed approach using RCI tests.

We report micro-average for precision and recall in Tabs. 7 and 8 for a naive approach (i.e., CUT-based RCI tests with a majority vote rule and a simple sequential strategy to resolve conflicts among orientations.) and our approach (i.e., the proposed RCI tests with the weak dependence detection mechanism, the majority vote rule and a maximal non-conflicting orientations strategy), respectively. The differences in both precision and recall between the two approaches are due to the effectiveness of our proposed RCI tests (as shown in the main text) and the fact that finding a maximally non-conflicting orientations works as a majority vote rule for final orientations of relational dependencies.

DETECTING CONFLICTS FOR RBO AND NON-RBO We investigate how weak dependency detection mechanisms for RBO and non-RBO work. In Fig. 6, we illustrate the average number of RCI tests which turned out to be colliders or non-colliders, and whether the RCI test results were right or wrong.

Without detection (the top row), we observe that there exists a non-negligible amount of wrong collider test results. This implies that a set of conditionals without blocking $\tilde{P}.Y$ (or $\tilde{Q}.Y$) yields wrong independence. This, again, suggests how false negatives dominate the performance of the learning algorithm.

With the detection mechanism enabled, the middle row

in the figure shows the average orientation results only when an empty set as a separating set is considered. Black bars represent cases where a pair of tests turned out to be dependent, that is, an orientation was not determined. Gray bars (nearly invisible) show cases where both tests returned independence. We can clearly see that the mechanism catches colliders better than without it.

The last row in the figure illustrates orientation results for the undetermined in the previous case (black bars). Note that, since the algorithm seeks for more than one separating set, the lengths of bars in the last row are longer than the lengths of black bars in the middle row. Collider-fail represents a condition where the detection mechanism rejects a collider since both tests yield independence. More than a half of cases, the mechanism correctly rejected false colliders, yielding a relatively low false collider rate.