Journal of Artificial Intelligence Research 47 (2013) 649-695 Submitted 02/13; published 08/13

Protecting Privacy through Distributed Computation in Multi-agent Decision Making

Online Appendix 1: Variable Election Algorithm

Thomas Léauté THOMAS.LEAUTE@A3.EPFL.CH
Boi Faltings BOI.FALTINGS@EPFL.CH

Algorithm 1, originally proposed in (Faltings, Léauté, & Petcu, 2008), elects one variable per connected component of the constraint graph, to serve as the root of a pseudo-tree. The algorithm consists in assigning a large, random score to each variable (line 1), such that scores are large enough to make collisions very improbable, and then electing the variable with the highest score, using a viral propagation mechanism (lines 10 to 16). If a simple viral propagation were used, each variable would be able to observe the converging sequence of tentative maximum scores received from each neighbor, and to infer the distance between that neighbor and the elected root, and also whether that neighbor has other neighbors. In order to prevent such inferences and guarantee topology privacy, the agents first lie a random, bounded number of times, by understating the true maximum score (lines 2 to 9).

To ensure termination and completeness, this algorithm assumes that agents know an upper bound $\phi_{\max}$ on the diameter of the constraint graph. If the agents did not lie, $\phi_{\max}$ iterations would be enough for the maximum score to propagate throughout the constraint

Algorithm 1 Anonymous root election algorithm for variable x.

Require: upper bound $\phi_{\max}$ on the constraint graph diameter
1: $score_x \leftarrow$ large random number
2: $max \leftarrow$ random number $\leq score_x$
3: $n_{lies} \leftarrow \text{rand}(\phi_{\max} \dots 2\phi_{\max})$
4: // Propagate under-estimates of the maximum score:
5: **for** n_{lies} times **do**
6: Send max to all neighbors
7: Get $max_1 \dots max_k$ from all neighbors
8: $max\_tmp \leftarrow \max(max, max_1, \dots, max_k)$
9: $max \leftarrow \text{rand}(max\_tmp \dots \max(score_x, max\_tmp))$
10: // Propagate the true maximum score:
11: $max \leftarrow \max(max, score_x)$
12: **for** $(3\phi_{\max} - n_{lies})$ times **do**
13: Send max to all neighbors
14: Get $max_1 \dots max_k$ from all neighbors
15: $max \leftarrow \max(max, max_1, \dots, max_k)$
16: **if** $max = score_x$ **then** x is the elected root

graph. To sufficiently obfuscate the viral propagation (see the proof of Theorem 2), the algorithm requires the agents to lie at least $\phi_{\max}$ and at most $2\phi_{\max}$ times; therefore, in total, $3\phi_{\max}$ iterations are sufficient for the algorithm to converge. Upon termination, only the elected variable knows that it is the root.

Theorem 1. *The root variable election procedure in Algorithm 1 terminates. Upon termination, only one variable (per connected component of the constraint graph) has been elected, with a probability that can be made arbitrarily close to* 1*.*

Proof. Algorithm 1 is a viral propagation procedure that is guaranteed to terminate in exactly $3\phi_{\max}$ steps, after which a single variable (per connected component of the constraint graph) has been elected, unless a collision occurs in which two or more variables have been assigned the maximum score (line 1). If scores are drawn uniformly from $[0, s_{\max})$, then the probability of such a collision can be made arbitrarily small by choosing $s_{\max}$ sufficiently large. The algorithm requires the exchange of exactly $6nn_e\phi_{\max} \in O(\phi \cdot d \cdot n^2)$ messages, where n is the number of variables, n_e is the number of edges, ϕ is the diameter of the constraint graph, and d its degree. □

Theorem 2. *Algorithm 1 guarantees full agent privacy, full constraint privacy, full decision privacy, and* partial *topology privacy. An upper bound on the diameter of the constraint graph is leaked, and a variable* might *also be able to discover:*

- *a lower bound on a neighbor variable's degree in the constraint graph;*
- *a lower bound on the total number of variables;*
- *bounds on the length of a cycle it is involved in.*

Proof. Constraint privacy and decision privacy are obviously fully guaranteed, since the algorithm does not even make use of any information about the feasibility values of the constraints in the problem.

Full agent privacy The messages exchanged only contain random numbers, which cannot be linked to any agent. In particular, the elected root variable is only revealed to the agent that owns it; other agents cannot discover the identity of this owner agent.

Partial topology privacy Conceptually, this algorithm uses a viral propagation mechanism to compute and reveal to all agents the maximum variable score across the constraint graph. Each variable's score is chosen as a large random number (line 1), so that revealing the maximum score to all agents does not reveal any topological information about the variable that has been assigned this maximum score. To decide when to stop, the algorithm assumes that the agents know an upper bound $\phi_{\max}$ on the constraint graph diameter, which is a minor leak of topology privacy.

During each round of this viral propagation procedure, each variable receives scores $max_1, \ldots, max_k$ from its k neighboring variables, and then replies with a score max at least equal to $\max(max_1, \ldots, max_k)$ (the same max is sent to all neighbors). By analyzing the history of scores received from its neighbors, an agent may be able to

make inferences about the topology of the constraint graph; Algorithm 1 includes a number of mechanisms to prevent this leak of topology privacy.

Let us first consider whether it is possible for a variable x to discover the existence of non-neighboring variables. In a pure viral propagation procedure, during the first round, x would receive the scores of all its neighbors. Any message received during a later round and containing a new, different score would reveal the existence of a non-neighboring variable with that score. To prevent such an inference, agents are required to under-report their respective scores during a positive, bounded number of rounds. More precisely, if an agent's score is greater than the maximum score it has seen in previous rounds, instead of reporting its true score, it sends to its neighbors a random number lower than its score. This number is still required to be greater than the maximum score seen so far, so that neighbors cannot detect that the agent lied. As long as its neighboring variable y could still be lying, x cannot conclude whether an unknown max received from y is due to a lie by y or whether y received it from a third variable at the previous round. However, to guarantee termination, agents can only be allowed to lie during a bounded number of rounds (in this algorithm, at most $2\phi_{\max}$ rounds). Therefore, after $(2\phi_{\max}+1)$ rounds, if x receives an unknown max from y, it will still discover that y has another neighbor, which cannot be a neighbor of x otherwise x would have received the max from that neighbor at the previous round. However, nothing else is leaked about the identity of that other variable.

Let us now consider whether a variable x can discover the existence of an edge in the constraint graph between two neighboring variables y and z, based on the respective max values it receives from them. The effect of the existence of an edge between y and z is that, if z sends $max_z^{(k)}$ at round k, x and y will both receive it at round $(k+1)$. Therefore, if x also receives $max_z^{(k)}$ from y at round $(k+2)$, x might be tempted to infer that y probably received it from z at iteration $(k+1)$, and therefore that y and z are neighbors. However this inference would be unfounded, because it is actually

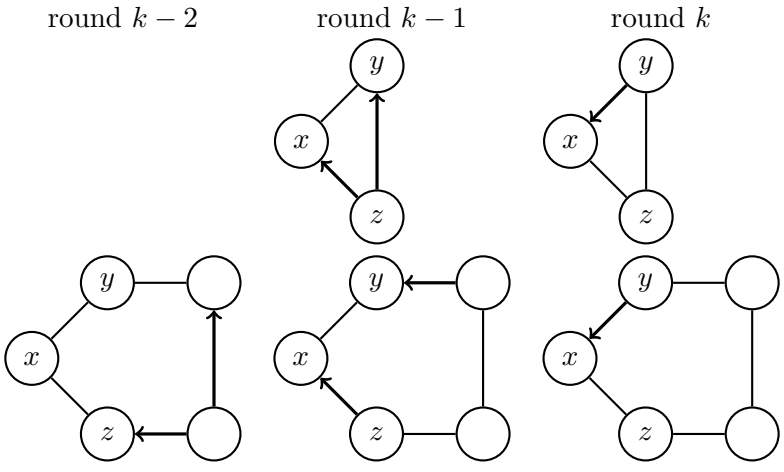

Figure 1: Two indistinguishable timelines from x's point of view.

equally probable that y and z are non-neighbors, and that $max_z^{(k)}$ was originally sent at round $j < k$ by another variable that is at distance $(k - j)$ from z and $(k - j + 1)$ from y. This is illustrated in Figure 1. To sum it up, x might be able to detect with some probability the existence of a cycle of length at most $2k+1$ involving x, y and z. In the most extreme case of $k = 1$, in this context x might be able to discover that y and z are neighbors. However, it might be possible to fix this privacy leak by having each agent report a *different* under-estimated max to each of its neighbors in a given round; the consequences of such a change to the algorithm remain to be investigated.

Finally, it is important to ask oneself whether the algorithm leaks any information about the *position* of the elected variable in the constraint graph. Notice that if such information were leaked, this would not necessarily be considered a violation of topology privacy (except if it revealed the existence of a non-neighbor variable); however, the knowledge of the position of the root of the pseudo-tree could be useful to the agents in order to make inferences during the following steps of the algorithm. In a pure viral propagation procedure, by observing the round at which the maximum score was received from each neighbor, an agent would be able to infer the distance (in number of edges) between that neighbor and the elected root. In particular, if the root were a neighbor, the agent would discover it. Algorithm 1 prevents this by requiring that agents lie at least $\phi_{\max}$ times, where $\phi_{\max}$ is an upper bound on the graph diameter, i.e. on the distance between any pair of variables. This way, the maximum score will only be received after at least $\phi_{\max}$ rounds, which therefore does not leak any useful information.

This concludes the proof of the privacy properties of Algorithm 1. □

References

Faltings, B., Léauté, T., & Petcu, A. (2008). Privacy guarantees through distributed constraint satisfaction. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'08)*, pp. 350–358.