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 3: Unique ID Generation Algorithm

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

The assignment of unique IDs to n variables is an instance of the well-known *renaming problem*, for which multiple algorithms have been proposed in the literature on distributed algorithms. However, to our knowledge, all these algorithms focus on robustness to failures, and ignore the issue of privacy. On the contrary, in this paper we do not consider agent failures, and we rather need an algorithm that protects agent and topology privacy. To this purpose, we propose Algorithm 1, which is a modification of the pseudo-tree generation algorithm in Online Appendix 2, and is an improved version of the algorithm proposed by Léauté and Faltings (2009). Each variable x is assigned a unique number id_x that corresponds to the order in which it is first visited during the distributed traversal of the constraint graph (or, more precisely, an upper bound thereon). This is done by appending to each CHILD message the number id of variables visited so far (lines 8, 29 and 31). Each variable adds a random number to id so as not to leak any useful upper bound on its number of neighbors (lines 5 and 15). At the end of this algorithm, the root variable discovers an upper bound n^+ on the total number of variables, and reveals it to everyone (lines 35 and 22 to 24).

Theorem 1. *The unique variable ID assignment algorithm guarantees full agent privacy, and partial topology privacy. The minor leaks of topology privacy lie in the fact that a variable* might *be able to discover:*

- *upper and lower bounds on the total number of variables;*
- *that there exists another branch in the constraint graph that it is not involved in.*

Proof. This is a modification of the depth-first traversal of the constraint graph in Online Appendix 2, which was shown to guarantee full agent privacy and full topology privacy. One difference is that the messages now also carry an integer id that is an upper bound on the number of variables visited so far, which cannot be used to make inferences about the identities of agents, but is a source of (minor) topology privacy leaks, as described below.

- When receiving a CHILD message from its parent, variable x discovers that there exist at most id other variables in the problem that have already been visited. These variables are either ancestors of its parent, or descendants of its parent in another branch of the pseudo-tree. To make sure this bound is loose and uninformative, each variable adds a random number in $[incr_{\min} \ldots 2incr_{\min}]$ to its id, where $incr_{\min}$ is a free parameter of the algorithm. In particular, this prevents the case $id = 1$, which

Algorithm 1 Pseudo-tree and unique ID generation algorithm for variable x

```
Require: a root variable
1:  if $x$ has at least one neighbor then
2:      $open_x \leftarrow \emptyset$
3:      if $x$ has been elected as the root then
4:          $id_x \leftarrow 0$
5:          $id_x^+ \leftarrow id_x + rand(incr_{\min} \ldots 2incr_{\min})$
6:          $open_x \leftarrow$ all neighbors of $x$
7:          Remove a random neighbor $y_0$ from $open_x$ and add it to $children_x$
8:          Send the message (CHILD, $id_x^+ + 1$) to $y_0$
9:      loop
10:         Wait for an incoming message ($type$, $id$) from a neighbor $y_i$
11:         if $open_x = \emptyset$ then // first time $x$ is visited
12:             $open_x \leftarrow$ all neighbors of $x$ except $y_i$
13:             $parent_x \leftarrow y_i$
14:             $id_x \leftarrow id$
15:             $id_x^+ \leftarrow id_x + rand(incr_{\min} \ldots 2incr_{\min})$
16:         else if $type$ = CHILD and $y_i \in open_x$ then
17:             Remove $y_i$ from $open_x$ and add it to $pseudo\_children_x$
18:             Send message (PSEUDO, $id$) to $y_i$
19:             next
20:         else if $type$ = PSEUDO then
21:             Remove $y_i$ from $children_x$ and add it to $pseudo\_parents_x$
22:         else if $type$ = NBRVARS then
23:             $n^+ \leftarrow id$ // upper bound on the true number of variables $n$
24:             break
25:         // Forward the CHILD message to the next $open$ neighbor:
26:         Choose a random $y_j \in open_x$
27:         if there exists such a $y_j$ then
28:             Remove $y_i$ from $open_x$ and add it to $children_x$
29:             Send the message (CHILD, $\max(id, id_x^+ + 1)$) to $y_j$
30:         else if $x$ is not the elected root then // backtrack
31:             Send message (CHILD, $\max(id, id_x^+ + 1)$) to $parent_x$
32:         else
33:             $n^+ \leftarrow id$ // upper bound on the true number of variables $n$
34:             break
35:     Send message (NBRVARS, $n^+$) to all children of $x$
```

would allow x to infer that its parent is the root of the pseudo-tree, and that x is its first child. Variable x also discovers that there exist at least $\lfloor id/(2incr_{\min})\rfloor$ already-visited variables. Notice that these two bounds are not informative about the topology of the constraint graph, since the $\mathrm{P}^{3/2}$-DPOP$^{(+)}$ and P^2-DPOP$^{(+)}$ algorithms leak the total number of variables anyway.

- When receiving a CHILD message from a child y, variable x can compare the *id* it contains with the *id* it previously sent to y. The difference $\Delta id \geq incr_{\min}$ is an upper bound on the number of y's descendants; the parameter $incr_{\min}$ can be chosen as large as necessary to make this bound as loose as desired. Variable x also discovers that $\lfloor \Delta id/(2incr_{\min}) \rfloor$ is a lower bound on the number of y's descendants.
- The *id* contained in a PSEUDO message does not provide any information, as it is equal to the *id* in the CHILD message to which the PSEUDO message is a response. In fact, *id* could be removed from the PSEUDO message; it has only be left in Algorithm 1 for the sake of conciseness of the pseudo-code.

Once all variables have been assigned unique IDs, the last such ID is revealed to all variables (and is recorded as n^+). This reveals an upper bound on the total number of variables, which is useless since the exact total number of variables is later revealed anyway by the $\mathrm{P}^{3/2}$-DPOP$^{(+)}$ and P^2-DPOP$^{(+)}$ algorithms. However, it also reveals to each variable whether it is the last visited variable, i.e. the last leaf of the pseudo-tree. This is another, minor leak of topology privacy, since the leaves of the pseudo-tree that are not the last leaf discover that there exists at least one other branch in the pseudo-tree (and therefore in the constraint graph) that they are not involved in. □

References

Léauté, T., & Faltings, B. (2009). Privacy-preserving multi-agent constraint satisfaction. In *Proceedings of the 2009 IEEE International Conference on PrivAcy, Security, riSk And Trust (PASSAT'09)*, pp. 17–25.