Solving Gossip Problems using Answer Set Programming: An Epistemic Planning Approach

Esra Erdem, Andreas Herzig

▶ **To cite this version:**

Esra Erdem, Andreas Herzig. Solving Gossip Problems using Answer Set Programming: An Epistemic Planning Approach. Electronic Proceedings in Theoretical Computer Science, 2020, Proceedings 36th International Conference on Logic Programming (Technical Communications), 325, pp.52-58. 10.4204/EPTCS.325.11 . hal-03012105

HAL Id: hal-03012105
https://hal.science/hal-03012105v1

Submitted on 18 Nov 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

Solving Gossip Problems using Answer Set Programming: An Epistemic Planning Approach

Esra Erdem
Sabanci University, Faculty of Engineering and Natural Sciences, Istanbul, Turkey
esraerdem@sabanciuniv.edu

Andreas Herzig
Institut de Recherche en Informatique de Toulouse, Université Paul Sabatier, Toulouse, France
Andreas.Herzig@irit.fr

We investigate the use of Answer Set Programming to solve variations of gossip problems, by modeling them as epistemic planning problems.

1 Introduction

The gossip problem is described by Hedetniemi et al. in their survey [10] as follows:

> Gossiping refers to the information dissemination problem that exists when each member of a set A of n individuals knows a unique piece of information and must transmit it to every other person. The problem is solved by producing a sequence of unordered pairs (i, j), $i, j \in A$, each of which represents a phone call made between a pair of individuals, such that during each call the two people involved exchange all of the information they know at that time; and such that at the end of the sequence of calls, everybody knows everything. Such a calling sequence, which completes gossiping among the n people, is called *complete*.

The gossip problem has been studied by many researchers, in particular, in the context of communication networks. While the most widely studied variant is the following optimization problem:

M Minimize the number of calls in a complete calling sequence.

there are other interesting variants considering the communication network and the sharing of gossips:

C Parallel communications: Concurrency of calls is allowed, but not two calls to and/or from the same agent at the same time.

D Directional gossips: Communication is limited from some agents to some other agents; it does not have to be bi-directional.

N Negative goals: Some agents should not know the secrets of some other agents in the end.

The variations **D** and **N** of the gossip problem are intractable [11, Section 4] [6, Theorem 10]. Optimal protocols for these variants are studied in [7].

We are also interested in the following variation of the gossip problem, from the perspective of multiagent epistemic reasoning:

E Higher-order epistemic goals: Some agents should know that some other specified agents know that ... that some other agents know some secrets in the end.

doi:10.4204/EPTCS.325.11

© Erdem and Herzig
This work is licensed under the
Creative Commons Attribution License.

Other variants exist, for example dynamic gossip where the communication graph evolves, which can be thought of as agents communicating telephone numbers of other agents during a call [8]; we donot study these dynamic variants in this paper.

We consider the mathematical definitions of the gossip problems as in [6], and illustrate how they can be solved using Answer Set Programming (ASP) [2] in the spirit of epistemic planning [5] where the gossip problem and its variants can be viewed as a paradigmatic problem that provides several benchmarks. Secrets are viewed as propositions that are either true or false.

2 Representing and Solving Gossip Problems in ASP

We represent the gossip problem and its variations as epistemic planning problems in the ASP language ASP-Core-2 [3], and utilize the ASP solver CLINGO [9] to compute solutions.

In ASP-Core-2, the object/predicate constants are denoted by strings that start with lower-case letters, and variables are denoted by strings that start with upper-case letters, very much like in Prolog.

2.1 Domain predicates

The atemporal input of the gossip problem are represented by atoms of the forms

- `agent(i)` – `i` denotes an agent,
- `secret(x)` – `x` denotes a secret that may be known and shared by agents,
- `connected(i,j)` – agent `i` can pass a secret to agent `j`.

These atoms are used to specify the domains of variables used in the formulas, and thus they are called "domain predicates".

If there are `n` agents and `s` secrets then these atoms can be represented in a more compact as follows:

```
agent(1..n).
secret(1..s).
```

For a complete communication network, where every agent can pass their secrets to every other agent, `connected(i,j)` can be defined as follows:

```
connected(I,J) :- agent(I), agent(J), I!=J.
```

For variation **D** of the gossip problem, if the communication network is not complete, `connected(i,j)` can be defined differently.

2.2 Auxiliary definitions: Pieces of information

Secrets and pieces of information that the agents know and can communicate to each other are defined recursively with respect to an epistemic depth.

The epistemic depth is specified by atoms of the form `depth(y)` as follows:

```
depth(0..d).
```

where the constant `d` specifies the maximum depth.

As the base case, at epistemic depth 0, every secret `K` can be a piece of information that an agent knows (i.e., the agent knows-whether `K`).

```
info(K,0) :- secret(K).
```

Intuitively, `info(K,0)` expresses that `K` is a piece of information of depth 0.

At epistemic depth 1, we consider pieces of information of the form "an agent `I` knows-whether `K`".

```
info(kw(I,K),1) :- agent(I), info(K,0).
```

Here `kw(I,K)` corresponds to $\mathbf{Kw}_I K$ of DL-PA [1] as adapted by Cooper et al. [5].

At an epistemic depth `D` greater than 1, pieces of information are nested like "an agent `I` knows-whether agent `J` knows-whether `K`", and can be defined as follows:

```
info(kw(I,kw(J,K)),D) :- agent(I;J), I!=J, info(kw(J,K),D-1), depth(D), D>1.
```

Since we are interested in solving gossip problems as planning problems, introspective statements are, in a sense, irrelevant; this is why `I!=J` in the body of the rule above.

2.3 Fluents: Knowledge of agents

The temporal input/output/constraints of the gossip problem (e.g., which agent knows-whether or should know-whether which secret, initially, at some time step, or in the end) can be represented with fluent constants – atoms of the form

`kww(i,x,t)` – agent `i` knows-whether secret `x` at step `t`.

Time steps, starting from `0` to a given upper bound `m` on makespan of plans, are specified as follows:

```
time(0..m).
```

Initial values of fluents Initially, the agents may already know-whether some secrets `K` (at epistemic depth `0`):

```
{kww(I,info(K,0),0)} :- agent(I), info(K,0).
```

as well as some pieces of information about themselves or other agents (at epistemic depth `D`):

```
{kww(I,info(kw(J,K),D),0)} :- agent(I), I!=J, info(kw(J,K),D).
```

Further conditions can be presented about the initial state by means of constraints, as illustrated by the following examples.

- Initially, every agent `I` knows-whether secret `I`:

  ```
  :- not kww(I,info(I,0),0), agent(I), info(I,0).
  ```

  and they do not know any other secrets, or pieces of information of epistemic depth greater than 0:

  ```
  :- kww(I,info(K,0),0), agent(I), info(K,0), I!=K.
  :- kww(I,info(K,D),0), agent(I), info(K,D), D>0.
  ```

  Cooper et al. ([5, Example 2]) consider the initial state of the gossip problem, as in the example above: $s_0 = \{\mathbf{Kw}_i s_i : 1 \leq i \leq n\}$.

- Initially, Agent 1 knows-whether Agent 2 knows-whether secret 2:

  ```
  :- not kww(1,info(kw(2,2),1),0).
  ```

- Initially, every agent knows-whether at least `l` secret:

  ```
  :- not l {kww(I,info(K,0),0) : info(K,0)}, agent(I).
  ```

  and/or at most `u` secrets:

  ```
  :- {kww(I,info(K,0),0) : info(K,0)} u, agent(I).
  ```

- Initially, no two agents know-whether the same piece of information:

  ```
  :- 2 {kww(I,info(K,_),0): agent(I)}, info(K,_).
  ```

Goal values of fluents Goal conditions can be represented by constraints as well, as illustrated in the following examples.

- Every agent knows-whether at least `N` pieces of information:

  ```
  goal(T) :-
    N {kww(I,info(K,0),T): agent(I), info(K,0) ;
       kww(I,info(K,D),T): agent(I), info(K,D), depth(D), D>0},
    infoNo(N), time(T).
  ```

  The secrets are of epistemic depth `0` (i.e., `info(K,0)`), whereas other pieces of information are of larger epistemic depths (i.e., `info(K,D)` where `D>0`).

  The number `N` can be specified as a constant, e.g., by a fact, as follows:

  ```
  infoNo(2).
  ```

  Alternatively, it can be defined as the total number of pieces of information of maximum epistemic depth `d`, for introspective `n` agents and `s` unique secrets, as follows:

  ```
  infoNo(N) :- infoNoAux(N,_,d).
  infoNoAux(s,s,0).
  infoNoAux(n*N+N1,n*N,D+1) :- infoNoAux(N1,N,D), depth(D), depth(D+1).
  ```

  Here, auxiliary atoms of the form `infoNoAux(N1,N,D)` represent the total number `N1` of pieces of information of maximum epistemic depth `D`, provided that the total number of pieces of information of epistemic depth `D-1` is `N`. The second line above expresses that, the number of secrets that might be known by the agents is `s`. The third line defines the total number of pieces of information over epistemic depths `D>0` recursively. If the number of pieces of information of epistemic depth `D` is `N`, then the maximum number of pieces of information of epistemic depth `D+1` is `n*N`. Therefore, if the total number of pieces of information of maximum epistemic depth `D` is `N1`, then the total number of pieces of information of maximum epistemic depth `D+1` is `n*N+N1`.

- Some agents should know-whether some secrets but not know-whether some other secrets. For instance, the following goal conditions express that agent `1` should know-whether secrets `1` and `3` but not secret `2`:

  ```
  goal(1,T) :- kww(1,info(1,0),T), info(1,0),
    kww(1,info(3,0),T), info(3,0),
    not kww(1,info(2,0),T), info(2,0), time(T).
  ```

  whereas other agents `I` know-whether all `s` messages:

  ```
  goal(I,T) :- agent(I), I!=1,
    s {kww(I,info(K,0),T): info(K,0)}, time(T).
  ```

  The goal is reached when these goal conditions hold at a time step `T`:

  ```
  goal(T) :- n {goal(I,T): agent(I)}, time(T).
  ```

  The variation **N** of the gossip problem has negative goal conditions, which can be specified as in the example above.

Once the goal conditions are defined, we ensure that the plan reaches a goal state at some time step `T`:

```
goal :- goal(T).
:- not goal.
```

Persistence of values of fluents Since the gossip problem does not involve forgetting, if an agent knows-whether a piece of information then he keeps that information:

```
kww(I,info(K,D),T+1) :- kww(I,info(K,D),T), agent(I), info(K,D), time(T), T<m.
```

2.4 Actions: Calls between agents

The output of the gossip problem is characterized by action constants – atoms of the form `call(i,j,t)` ("agent `i` calls agent `j` at time step `t`.")

Action occurrences and preconditions Agent `I` is free to call agent `J` at any time, if he is allowed to (relative to the communication network):

```
{call(I,J,T)} :- agent(I), agent(J), time(T), I!=J, connected(I,J), T<m.
```

Effects of actions When an agent `I` calls agent `J` at time step `T`, all permitted pieces of information `K` of `I` are passed to `J`:

```
kww(J,info(K,D),T+1) :- call(I,J,T), agent(I), agent(J),
  kww(I,info(K,D),T), info(K,D), permitted(I,J,K,T), time(T), T<m.
```

Furthermore, agent `J` knows-whether agent `I` knows-whether information `K`:

```
kww(J,info(kw(I,K),D+1),T+1) :- call(I,J,T), agent(I), agent(J), info(K,D),
  kww(I,info(K,D),T), info(kw(I,K),D+1), permitted(I,J,K,T), time(T), T<m.
```

As in the original gossip problem, as an effect of `call(I,J,T)`, all permitted pieces of information `K` of `J` are passed to `I`:

```
kww(I,info(K,D),T+1) :- call(I,J,T), agent(I), agent(J),
  kww(J,info(K,D),T), info(K,D), permitted(J,I,K,T), time(T), T<m.
kww(I,info(kw(J,K),D+1),T+1) :- call(I,J,T), agent(I), agent(J), info(K,D),
  kww(J,info(K,D),T), info(kw(J,K),D+1), permitted(J,I,K,T), time(T), T<m.
```

Permitted messages It can be assumed that all pieces of information are permitted to be shared at any time, unless told otherwise:

```
permitted(I,J,K,T) :- connected(I,J), agent(I), agent(J), info(K,_),
  kww(I,info(K,_),T), not -permitted(I,J,K,T), time(T), T<m.
```

Alternatively, agents can pass at least `ll` and at most `uu` pieces of information at any time:

```
ll {permitted(I,J,K,T) : info(K,_), kww(I,info(K,_),T)} uu :-
  agent(I), agent(J), connected(I,J), time(T), T<m.
```

2.5 Constraints and preferences

Concurrency can be prevented to some extent. For instance, an agent cannot be called while making a call, or by more than one agent at the same time. Also, an agent cannot call more than one agent at the same time. These concurrency constraints can be represented as follows:

```
:- 2 {call(I,J,T): agent(J), connected(I,J);
  call(J1,I,T): agent(J1), connected(J1,I)}, agent(I), time(T), T<m.
```

Unless told otherwise, the formulation of calls as in the previous section, and with the constraint above, allows parallel communication as in the variation **C** of the gossip problem.

We can also express preferences about occurrences of actions. For instance, the following "weak" constraint expresses our preference of less number of calls (as in the variation **M** of the gossip problem):

```
:~ call(I,J,T), connected(I,J), agent(I;J), time(T), T<m. [3@1,I,J,T]
```

Intuitively, with these weak constraints, every atom of the form `call(I,J,T)` included in an answer set S costs 3 units for S. Then the total cost of an answer set characterizes to what extent the preferences are not satisfied in a plan: the larger the total cost, the less preferred the plan is. The ASP solver CLINGO finds an answer set whose total cost is minimum, and thus a plan with minimum number of calls.

The following weak constraint expresses our preference over the satisfaction of goal conditions at an earlier time step:

```
:~ not goal(T), time(T). [3@2,T]
```

Intuitively, if a goal is not satisfied at time step `T` in an answer set, then the total cost of the answer set is increased by 3. Therefore, with these weak constraints, the makespan of a plan is minimized. This optimization has a higher priority, i.e., of `2`, compared to the minimization of the number of calls. Therefore, CLINGO first minimizes the makespan, and then tries to minimize the number of calls.

2.6 Epistemic planning problem

For instance, for 4 agents, 4 messages, a complete communication network between agents, and an epistemic depth of 1, a solution to the gossip problem (with the conditions **M**, **C**) computed by the ASP solver CLINGO [9] for maximum time step 2 is as follows:

```
call(1,2,0) call(4,3,0) call(2,3,1) call(1,4,1)
```

For epistemic depth of 2, the computed solution for maximum time step 4 is as follows:

```
call(1,2,0) call(3,4,0) call(4,2,1) call(3,1,1)
call(4,1,2) call(4,2,3) call(1,3,3)
```

3 Evaluations

We have performed some experiments for the gossip problem instances with the conditions **M**, **C**, where the goal is specified with respect to the epistemic depth 1 and the plan lengths are also minimized. A complete communication network is considered between the agents. The maximum makespan is specified as 5.

In the experiments, we have used CLINGO 5.2.2, and performed experiments on a Linux server with 3.30GHz Intel(R) Xeon(R) W-2155 CPU and 32 GB memory. A maximum threshold of 10 seconds is provided to CLINGO, allowing anytime search.

The experimental results are presented in Table 1. As can be seen from the table, for `n`=2,3,4,5,6,8, the solutions computed by CLINGO are optimal with minimum number of calls; for `n`=7 and `n` larger than 8, the solutions computed by CLINGO by anytime search may not be optimal. If we increase the time threshold to 600 seconds, an optimal solution for `n`=7 is computed in 336.928 seconds. For `n`=11,13,15, slightly better solutions with 19, 24, 30 calls, respectively, are computed, but they may not be optimal.

Table 1: Experimental results: For `n` agents and the maximum time step `m=5`, a solution to the gossip problem is found with anytime search of CLINGO within 10 seconds. For each solution, whether it is optimal (O) or may be optimal (+), and the number of calls are reported.

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15
time (sec)	0.004	0.005	0.005	0.082	0.049	10	3.685	10	10	10	10	10	10	10
optimality	O	O	O	O	O	+	O	+	+	+	+	+	+	+
# calls	1	3	4	6	8	10	12	14	17	20	21	25	28	31

4 Discussions

Gossip problems pose an interesting set of computational problems for Answer Set Programming (ASP). From the perspective of representation, it is challenging to model variations of gossip problems in an elaboration tolerant way and utilizing the useful constructs of ASP. From the perspective of computational performance, as can be seen from the preliminary experimental evaluations, although the ASP approach fares much better than the PDDL approach [4], it is still not scalable for larger instances. Our ongoing studies involve extension of experimental evaluations to other variations of gossip problems, and comparisons with the PDDL approach.

References

[1] Philippe Balbiani, Andreas Herzig & Nicolas Troquard (2013): *Dynamic Logic of Propositional Assignments: A Well-Behaved Variant of PDL*. In: *Proc. of LICS*, pp. 143–152, doi:10.1109/LICS.2013.20.

[2] Gerhard Brewka, Thomas Eiter & Miroslaw Truszczynski (2011): *Answer set programming at a glance*. *Commun. ACM* 54(12), pp. 92–103, doi:10.1145/2043174.2043195.

[3] Francesco Calimeri, Wolfgang Faber, Martin Gebser, Giovambattista Ianni, Roland Kaminski, Thomas Krennwallner, Nicola Leone, Francesco Ricca & Torsten Schaub (2020): *ASP-Core-2 Input language format*. *Theory and Practice of Logic Programming* 20(2), p. 294309, doi:10.1017/S1471068419000450.

[4] Martin Cooper, Andreas Herzig, Frédéric Maris, Elise Perrotin & Julien Vianey (2020): *Lightweight Parallel Multi-Agent Epistemic Planning*. In: *Proc. of KR*.

[5] Martin C. Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris & Pierre Régnier (2016): *A Simple Account of Multi-Agent Epistemic Planning*. In: *Proc. of ECAI*, doi:10.3233/978-1-61499-672-9-193.

[6] Martin C. Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris & Pierre Régnier (2016): *Simple epistemic planning: generalised gossiping*. *CoRR* abs/1606.03244.

[7] Martin C. Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris & Pierre Régnier (2019): *The epistemic gossip problem*. *Discret. Math.* 342(3), pp. 654–663, doi:10.1016/j.disc.2018.10.041.

[8] Hans van Ditmarsch, Jan van Eijck, Pere Pardo, Rahim Ramezanian & François Schwarzentruber (2017): *Epistemic protocols for dynamic gossip*. *J. Appl. Log.* 20, pp. 1–31, doi:10.1016/j.jal.2016.12.001.

[9] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub & Marius Thomas Schneider (2011): *Potassco: The Potsdam Answer Set Solving Collection*. *AI Commun.* 24(2), pp. 107–124, doi:10.3233/AIC-2011-0491.

[10] Sandra M. Hedetniemi, Stephen T. Hedetniemi & Arthur L. Liestman (1988): *A survey of gossiping and broadcasting in communication networks*. *Networks* 18(4), pp. 319–349, doi:10.1002/net.3230180406.

[11] David W. Krumme, George Cybenko & K. N. Venkataraman (1992): *Gossiping in Minimal Time*. *SIAM J. on Computing* 21, pp. 111–139, doi:10.1137/0221010.