Benchmark Problems for the Context Dependent Learning from Ordered Answer Sets Framework

Mark Law, Alessandra Russo, Krysia Broda
Department of Computing, Imperial College London, SW7 2AZ
(*e-mail:* {mark.law09, a.russo, k.broda}@imperial.ac.uk)

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Abstract

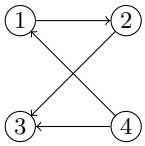
In this document, we present the detailed descriptions of the benchmark problems that were discussed in section 5 of the paper *Iterative Learning of Answer Set Programs from Context Dependent Examples*. We also give details of how to download and run these learning tasks with our own systems.

KEYWORDS: Non-monotonic Inductive Logic Programming, Answer Set Programming, Iterative Learning

1 Hamiltonian Graphs

The two learning tasks *Hamilton A* and *Hamilton B* were aimed at learning how to decide whether a graph is Hamiltonian of not.

The four node Hamiltonian graph G in figure 1 can be represented by the set of facts F_G.



$$F_G = \begin{cases} \texttt{node(1..4).} \\ \texttt{edge(1,2).} \\ \texttt{edge(2,3).} \\ \texttt{edge(3,4).} \\ \texttt{edge(4,1).} \end{cases}$$

Fig. 1: An example of a Hamiltonian Graph G and its corresponding representation in ASP, F_G.

To decide whether a graph is Hamiltonian or not, we can use the program H below:

```
reach(V0) :- in(1,V0).
reach(V1) :- in(V0,V1), reach(V0).
0 {in(V0,V1) } 1 :- edge(V0, V1).
:- node(V0), not reach(V0).
:- in(V0,V1), in(V0,V2), V1 != V2.
```

If for a graph G and its corresponding set of facts F_G, G is Hamiltonian if and only if $F_G \cup H$ is satisfiable. In Hamilton A and Hamilton B, we give two alternative encodings of a task which aims to learn H: the first in ILP_{LOAS} and the second in $ILP^{context}_{LOAS}$. Both learning tasks have the same hypothesis space, consisting of 104 normal rules, choice rules and hard constraints.

1.1 Hamilton A: non-context-dependent examples

In our ILP_{LOAS} representation of the learning task, the background B knowledge needed to generate the possible graphs of size $1, \ldots, 4$. B is as follows.

```
1 { node(1); node(2); node(3); node(4) } 4.
0 { edge(N1, N2) } 1 :- node(N1), node(N2).
```

Our examples then contained details of which edges were in (and not in) the graph. For example, one positive example was: $\langle\{$`node(1)`,`node(2)`,`edge(1,2)`, `edge(2,1)`$\}, \{$`node(3)`,`node(4)`,`edge(1,1)`,`edge(2,2)`$\}\rangle$.

The full encoding can be downloaded from `https://www.doc.ic.ac.uk/~ml1909/ILASP/benchmarks/hamiltonA.las` and can be run with the commands:

- `ILASP hamiltonA.las` (for ILASP2).
- `ILASP --2i hamiltonA.las` (for ILASP2i).
- `ILASP --2i -pt hamiltonA.las` (for ILASP2i with pre-translation).

1.2 Hamilton B: context-dependent examples

In the Hamilton B task, we made use of the context-dependent examples in $ILP^{context}_{LOAS}$. We therefore didn't need any background knowledge (it was empty), and instead encoded the graphs in the contexts of examples such as in the positive example: $\langle\langle\emptyset, \emptyset, \{$`node(1..2). edge(1,2). edge(2,1).`$\}\rangle$

The full encoding can be downloaded from `https://www.doc.ic.ac.uk/~ml1909/ILASP/benchmarks/hamiltonB.las` and can be run with the commands:

- `ILASP hamiltonB.las` (for ILASP2).
- `ILASP --2i hamiltonB.las` (for ILASP2i).
- `ILASP --2i -pt hamiltonB.las` (for ILASP2i with pre-translation).

2 Scheduling

In this problem setting, we are aiming to learn an academic's preferences about interview timetables for students. The timetables are over 9, 12 and 15 slots, corresponding to 3 day, 4 day and 5 day timetables, respectively. There are two courses: Computing and Joint Mathematics and Computing. In these three learning tasks, we aim to learn the preferences:

```
:~ assigned(V0, V1), assigned(V2, V1), neq(V0, V2).[2@1, 1, V0, V1, V2]
:~ assigned(V0, V1), type(V0,V1,jmc).[1@2, 2, V0, V1]
```

In the case of the 3 day timetable setting (Scheduling A), the background knowledge is as follows:

```
slot(1..3,1..3).

neq(1, 2).  neq(1, 3).  neq(2, 1).
neq(2, 3).  neq(3, 1).  neq(3, 2).

type(1,1,jmc).  type(1,2,c).    type(1,3,jmc).
type(2,1,c).    type(2,2,c).    type(2,3,c).
type(3,1,c).    type(3,2,jmc).  type(3,3,c).

0 { assigned(X, Y) } 1 :- slot(X,Y).
```

The background knowledge generated the answer sets corresponding to each possible timetable. Each (non-context-dependent) positive example then contained a partial description of a possible timetable, such as $\langle\{\texttt{assigned}(2,3), \texttt{assigned}(3,2), \texttt{assigned}(2,1), \texttt{assigned}(1,1)\}, \{\texttt{assigned}(2,2), \texttt{assigned}(3,3), \texttt{assigned}(1,3), \texttt{assigned}(3,1)\}\rangle$. The task then contained many brave and cautious orderings over these positive examples. In each case, the hypothesis space consisted of 180 possible weak constraints.

The full encoding can be downloaded from `https://www.doc.ic.ac.uk/~ml1909/ILASP/benchmarks/schedulingA.las`, `https://www.doc.ic.ac.uk/~ml1909/ILASP/benchmarks/schedulingB.las` and `https://www.doc.ic.ac.uk/~ml1909/ILASP/benchmarks/schedulingC.las` and can be run with the commands (for scheduling A):

- `ILASP schedulingA.las` (for ILASP2).
- `ILASP --2i schedulingB.las` (for ILASP2i).
- `ILASP --2i -pt schedulingC.las` (for ILASP2i with pre-translation).

3 Agent

In this scenario, an agent must learn how to navigate a grid. It starts with complete knowledge of the map, but no knowledge of which moves it will be able to make in future time points. At each time point, the agent is informed of which moves it can make by an oracle.

In these learning tasks, each of our examples is of a path taken by the agent through the maze, and which moves it was allowed to make at each time point. The rules which should be learned are different in each scenario, and so for some scenarios, different types of examples are needed.

3.1 Agent Scenario A

In this scenario, the agent must learn the following definition of valid move.

```
valid_move(C1, T) :-
  adjacent(C1, C2), agent_at(C2, T),
  unlocked(C1, T), not wall(C1, C2).
```

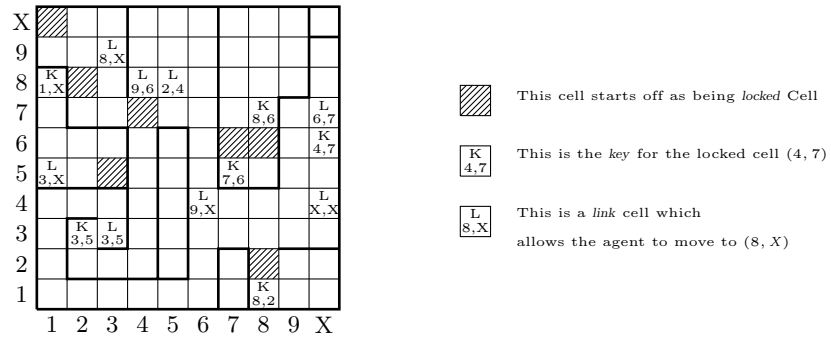


Fig. 2: Cells with diagonal lines are *locked* and the agent must visit the corresponding key before it can enter these cells. *Link* cells allow the agent to jump to the indicated destination cell. The thick black lines represent walls.

```
valid_move(C1, T) :-
  link(C2, C1), agent_at(C2, T),
  unlocked(C1, T).
```

The hypothesis space consists of 531 normal rules. The background knowledge contains the description of the map and there are only positive examples in this task. Each positive example has a context containing the agent's history (its path through the maze so far) and the inclusions and exclusions are moves which are valid and invalid, respectively; for instance, $\langle\{\texttt{valid\_move}(\texttt{cell}(9,1),1),\ldots\}, \{\texttt{valid\_move}(\texttt{cell}(1,3),1),\ldots\}\rangle$, $C\rangle$, where C is the set of facts:

```
agent_at(cell(10,1),1).   agent_at(cell(9,1),2).   agent_at(cell(8,1),3).
agent_at(cell(8,2),4).    agent_at(cell(8,3),5).   agent_at(cell(8,4),6).
agent_at(cell(7,4),7).    agent_at(cell(6,4),8).   agent_at(cell(9,10),9).
agent_at(cell(8,10),10).  agent_at(cell(8,9),11).  agent_at(cell(8,8),12).
agent_at(cell(8,7),13).
```

Note that this demonstrates the simplicity of $ILP^{context}_{LOAS}$ in giving different contexts to different examples. We are able to give the set of valid moves for each example history. The only way in ILP_{LOAS} to do this would be to encode the example histories as part of the background knowledge.

The full encoding can be downloaded from `https://www.doc.ic.ac.uk/~ml1909/ILASP/benchmarks/agentA.las`, and run with the commands:

- `ILASP agentA.las -ml=5 -nc` (for ILASP2).
- `ILASP --2i agentA.las -ml=5 -nc` (for ILASP2i).
- `ILASP --2i -pt agentA.las -ml=5 -nc` (for ILASP2i with pre-translation).

Note the extra arguments; `-nc`, means not to generate constraints in the hypothesis space, and `-ml=5` sets the maximum number of literals in a rule to 5, rather than ILASP's default of 4.

3.2 Agent Scenario B

In this scenario, the agent must learn the following definition of valid move:

```
valid_move(V0, V1) :- extra(V0, V1), visited_cell(V2, V1), key(V2, V0).
valid_move(V0, V1) :- extra(V0, V1), not locked(V0).
extra(V2, V1) :- agent_at(V0, V1), link(V0, V2).
extra(V2, V1) :- agent_at(V0, V1), not wall(V0, V2), adjacent(V0, V2).
```

This involves "inventing" the predicate `extra`. The task uses a hypothesis space of 146 normal rules. The 50 positive examples in this task are of the same structure as those in `agentA`. The full encoding can be downloaded from `https://www.doc.ic.ac.uk/~ml1909/ILASP/benchmarks/agentB.las`, and run with the commands:

- `ILASP agentB.las -nc` (for ILASP2).
- `ILASP --2i agentB.las -nc` (for ILASP2i).
- `ILASP --2i -pt agentB.las -nc` (for ILASP2i with pre-translation).

3.3 Agent Scenario C

In this scenario, the agent must learn the following definition of valid move:

```
valid_move(C1, T) :-
  adjacent(C1, C2), agent_at(C2, T),
  unlocked(C1, T), not wall(C1, C2).

valid_move(C1, T) :-
  link(C2, C1), agent_at(C2, T),
  unlocked(C1, T).

:- agent_at(C, T), already_visited_cell(C, T).
```

To do so, it needs positive examples of the same form as those in `agentA` and `agentB`, but as the hypothesis contains a constraint, it also needs negative examples, which rule out agent histories where the agent has visited the same cell twice. The concept of `already_visited_cell` is part of the background knowledge. The negative examples are similar to the positive examples, but contain invalid agent histories in the context. One such example is:

```
< < { valid_move(cell(9,1),1), ... }, { valid_move(cell(3,3),1), ... } >, {
  agent_at(cell(10,1),1).
  agent_at(cell(9,1),2).
  agent_at(cell(8,1),3).
  agent_at(cell(8,2),4).
  agent_at(cell(8,3),5).
  agent_at(cell(7,3),6).
  agent_at(cell(7,4),7).
  agent_at(cell(7,3),8).
  agent_at(cell(7,4),9).
  agent_at(cell(6,4),10).
  agent_at(cell(9,10),11).
  agent_at(cell(8,10),12).
  agent_at(cell(7,10),13).
  agent_at(cell(7,9),14).
}>
```

The hypothesis rule consists of 160 normal rules and hard constraints. The full encoding can be downloaded from `https://www.doc.ic.ac.uk/~ml1909/ILASP/benchmarks/agentC.las`, and run with the commands:

- `ILASP agentC.las -ml=5` (for ILASP2).
- `ILASP --2i agentC.las -ml=5` (for ILASP2i).
- `ILASP --2i -pt agentC.las -ml=5` (for ILASP2i with pre-translation).

3.4 Agent Scenario D

In this scenario, the agent must learn the following definition of valid move:

```
valid_move(V2, V1) :-
  agent_at(V0, V1), unlocked(V2, V1),
  link(V0, V2).

valid_move(V1, V2) :-
  adjacent(V0, V1), agent_at(V0, V2),
  unlocked(V1, V2), not wall(V0, V1).

:- agent_at(V0, V1), already_visited_cell(V0, V1).

:~ agent_at(V0, V1), not shielded(V1), danger_rating(V0, V2).[V2@3, 1, V0, V1, V2]
:~ agent_at(V0, V1), coin(V0, V2).[V2@2, 2, V0, V1, V2]
:~ agent_at(V0, V1).[1@1, 3, V0, V1]
```

The learning task contained similar positive and negative examples to `Agent C`, but also needed brave ordering examples to learn the weak constraints. The ordering examples were over the previously described positive examples, and described agent histories which were preferred (given the weak constraints) to other agent histories.

The full encoding can be downloaded from `https://www.doc.ic.ac.uk/~ml1909/ILASP/benchmarks/agentD.las`, and run with the commands:

- `ILASP agentD.las -ml=5` (for ILASP2).
- `ILASP --2i agentD.las -ml=5` (for ILASP2i).
- `ILASP --2i -pt agentD.las -ml=5` (for ILASP2i with pre-translation).

4 Journey Preferences

This learning task is an example similar to the tasks in the case study near the end of section 5 in the main paper. The aim for this task was to learn the following 3 weak constraints:

```
:~ leg_mode(L, walk), leg_crime_rating(L, C), C > 3.[1@3, L, C]
:~ leg_mode(L, car).[1@2, L]
:~ leg_mode(L, walk), leg_distance(L, D).[D@1, L, D]
```

We used context-dependent partial interpretations, each of which encoded the attributes of a single journey. One such example is:

```
< < {}, {} >, {
  leg_mode(leg(1), walk).
  leg_distance(leg(1), 1876).
  leg_crime_rating(leg(1), 2).

  leg_mode(leg(2), bus).
  leg_distance(leg(2), 5418).
  leg_crime_rating(leg(2), 3).

  leg_mode(leg(3), walk).
  leg_distance(leg(3), 92).
  leg_crime_rating(leg(3), 1).

  leg_mode(leg(4), bus).
  leg_distance(leg(4), 2842).
  leg_crime_rating(leg(4), 2).

  leg_mode(leg(5), walk).
  leg_distance(leg(5), 94).
  leg_crime_rating(leg(5), 2).

}>
```

The task then contained 200 brave orderings over these positive examples. The background knowledge (as in section 6 of the main paper) was empty, and the hypothesis space consisted of 117 weak constraints.

The full encoding can be downloaded from `https://www.doc.ic.ac.uk/~ml1909/ILASP/benchmarks/journey.las`, and run with the commands:

- `ILASP journey.las` (for ILASP2).
- `ILASP --2i journey.las` (for ILASP2i).
- `ILASP --2i -pt journey.las` (for ILASP2i with pre-translation).