

Deductive-Inductive Logic Programming for Collaborative Problem Solving

Jian Huang

433-482 Software Agents &
Agent Programming Languages

Semester 1, April 30 2008

Outline

- 1 Introduction to ILP
- 2 Integrating Deduction & Induction
- 3 Collaborative Problem Solving

Deduction Vs. Induction

Deductive Reasoning:

- Examples: Euclidean geometry, Prolog.
- Limitation: does not generalize.

Inductive Reasoning:

- Native to human.
- Central to scientific discovery.
- Limitation: error prone.

Reasoning $\left\{ \begin{array}{l} \text{Deductive reasoning} \\ \text{Inductive reasoning} \end{array} \right. \begin{array}{l} \rightarrow \text{LP} \\ \rightarrow \text{ILP} \end{array} \rightarrow \begin{array}{l} \text{Prolog} \\ \text{Aleph} \end{array}$

Inductive Reasoning

Inductive Reasoning is classified into:

- **Induction**: Inferring general rules from specific data
Example: swan example, marble example
- **Abduction**: Reasoning from effects to causes
Example: medical, detective, scientific

Aim:

- $B \wedge H \models E$

Inductive Logic Programming

Problem Formulation

Definition (Inductive Logic Programming)

Given B and $E = E^+ \cup E^-$ represented as logic programs, find H such that:

- 1 Necessity: $B \not\models E^+$
- 2 Sufficiency: $B \wedge H \models E^+$
- 3 Weak Consistency: $B \wedge H \not\models \square$
- 4 Strong Consistency: $B \wedge H \wedge E^- \not\models \square$

How to find H systematically?

Inductive Logic Programming

General Method

General method for finding hypothesis H :

Input: B , E^+ and E^- .

Output: H .

1. Start with some initial (possibly empty) H .
2. **repeat**
3. **if** $B \wedge H$ is too strong **then**
4. specialize H .
5. **if** $B \wedge H$ is too weak **then**
6. generalize H .
7. **until** all four conditions are met.
- 8.
9. **return** H .

Inductive Logic Programming

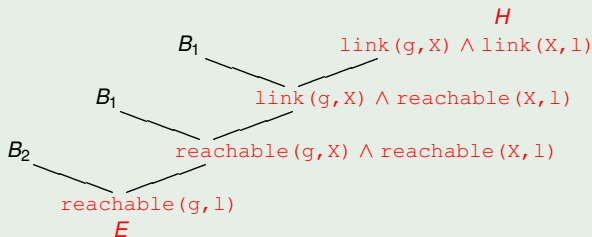
Inverse Resolution

Example

B_1 : $\text{link}(A, B) \rightarrow \text{reachable}(A, B)$

B_2 : $\text{reachable}(A, B) \wedge \text{reachable}(B, C) \rightarrow \text{reachable}(A, C)$

E : $\text{reachable}(g, l)$



H : $\text{link}(g, X) \wedge \text{link}(X, l)$, such that $B \wedge H \models E$

Outline

- 1 Introduction to ILP
- 2 Integrating Deduction & Induction
- 3 Collaborative Problem Solving

RichProlog

Limitation of deductive reasoning (e.g. Prolog)

- If A is a swan and is white, B is a swan and is white and C is a swan. What is the color of C?
- If A eats apple, banana, orange, grapes, Does A eat peach?

Combining deduction and induction

- RichProlog extends Prolog and answers this type of queries: Is there a pattern that matches these instances:

(aaa, aab, aba, abb)

or

$\exists X \forall Y \text{pattern}(X) \wedge \text{matches}(X, Y)$

Deductive-Inductive Logic Programming Framework

Notation

Let $\Sigma(T)$ represent deduction, $\Pi(T)$ represent induction

$$\Sigma_n(T) = \begin{cases} T & \text{if } n = 0 \\ \Sigma(\Pi_{n-1}(T)) & \text{if } n > 0 \end{cases}$$

$$\Pi_n(T) = \begin{cases} T & \text{if } n = 0 \\ \Pi(\Sigma_{n-1}(T)) & \text{if } n > 0 \end{cases}$$

The first few terms of $\Sigma_n(T)$ and $\Pi_n(T)$ are as follows:

$$\begin{array}{llll} \Sigma_0(T) & = & \Pi_0(T) & = & T & \text{– original theory} \\ \Sigma_1(T) & = & \Sigma(\Pi_0(T)) & = & \Sigma(T) & \text{– deduction on } T \\ \Pi_1(T) & = & \Pi(\Sigma_0(T)) & = & \Pi(T) & \text{– induction on } T \\ \Sigma_2(T) & = & \Sigma(\Pi_1(T)) & = & \Sigma(\Pi(T)) & \dots \\ \Pi_2(T) & = & \Pi(\Sigma_1(T)) & = & \Pi(\Sigma(T)) & \dots \\ \vdots & & & & & \end{array}$$

Deductive-Inductive Logic Programming Framework

Transformation Rules

Deductive-Inductive inference is a recursive application of these rules:

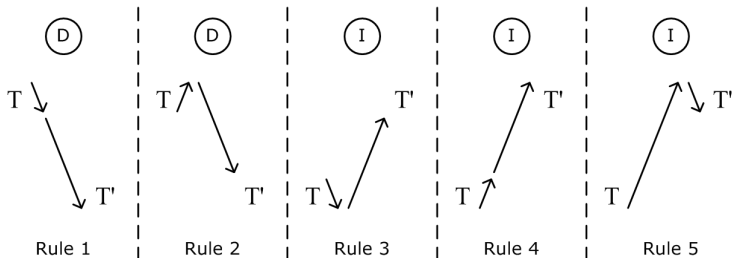
- 1 $\Sigma(T) \Rightarrow \Sigma(\Sigma(T))$ — Prolog
- 2 $\Sigma(T) \Rightarrow \Sigma(\Pi(T))$ — RichProlog (roughly)
- 3 $\Pi(T) \Rightarrow \Pi(\Pi(T))$
- 4 $\Pi(T) \Rightarrow \Pi(\Sigma(T))$
- 5 $\Pi(T) \Rightarrow \Sigma(\Pi(T))$ — RichProlog (roughly)

(RichProlog answers ONLY $\Sigma(\Pi(T))$ queries.)

Deductive-Inductive Logic Programming Framework

Transformation Rules

A graphical view of the transformation rules:



Deductive-Inductive Logic Programming Framework

An Example

Given the following knowledge, can we deduce $\text{reachable}(b, d)$?

Example (Reachability)

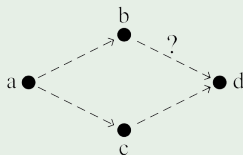
f_1 : $\text{reachable}(a, c)$

f_2 : $\text{reachable}(a, b)$

f_3 : $\text{reachable}(c, d)$

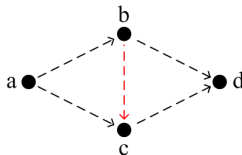
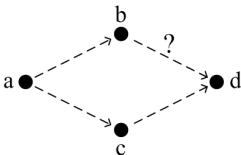
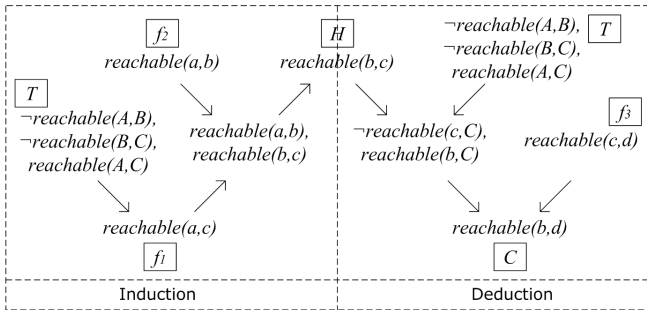
T : $\text{reachable}(A, B) \wedge \text{reachable}(B, C)$
 $\rightarrow \text{reachable}(A, C)$

Q : $?\text{reachable}(b, d)$



Deductive-Inductive Logic Programming Framework

An Example



Outline

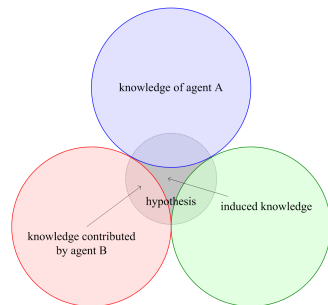
- 1 Introduction to ILP
- 2 Integrating Deduction & Induction
- 3 Collaborative Problem Solving**

Problem Solving in Multi-Agent Systems

Issues specific to multi-agent environments:

- Knowledge being distributed;
- Exposure of internal knowledge (cost, privacy, trust etc);
- Knowledge being incomplete;
- Pirates example.

Extend deductive-inductive inference to allow interaction.



Distributed path planning

Problem setting

Each agent has the following background knowledge:

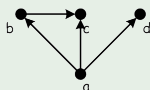
Example (background knowledge)

$$B_1 : \text{link}(A, B) \rightarrow \text{reachable}(A, B)$$

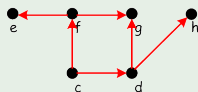
$$B_2 : \text{reachable}(A, B) \wedge \text{reachable}(B, C) \rightarrow \text{reachable}(A, C)$$

Plus, each agent knows the links it has travelled, e.g.:

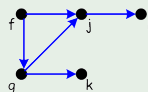
Example (background knowledge)



Car A



Car B

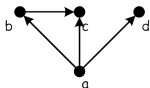
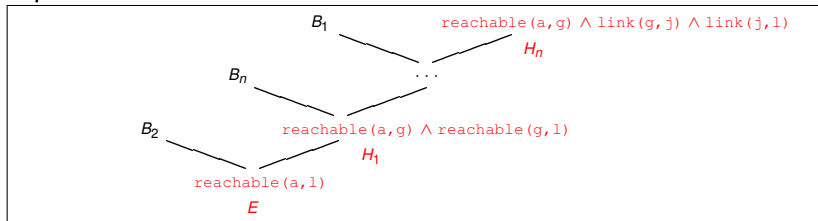


Car C

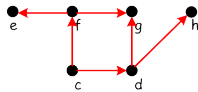
Distributed path planning using DILP

Inducing a path

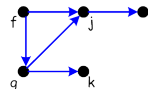
A path can be induced:



Car A



Car B

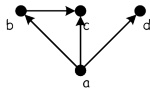


Car C

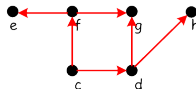
Distributed path planning using DILP

Collaboration

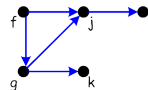
-
-
- 1 A : $E = \text{reachable}(a, l)$
 - 2 A ASKS C : $E = \text{reachable}(a, l)$
 - 3 C INDUCES: $H = \text{reachable}(a, g) \wedge \text{link}(g, j) \wedge \text{link}(j, l)$
 - 4 C REPLIES: $H = \text{reachable}(a, g)$
 - 5 A DEDUCES: $K_1 = K_a(K_c(\text{reachable}(g, l)))$
 $K_2 = K_a(\exists i K_i(\text{reachable}(a, g)) \rightarrow K_a(\text{reachable}(a, l)))$
 - 6 A : $E = \text{reachable}(a, g)$
-
-



Car A



Car B

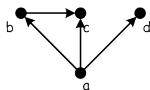


Car C

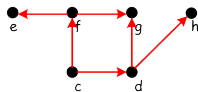
Distributed path planning using DILP

Collaboration

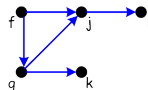
-
-
- 6 A : $E = \text{reachable}(a, g)$
- 7 A ASKS B : $E = \text{reachable}(a, g)$
- 8 B INDUCES: $H = \text{reachable}(a, c) \wedge \text{link}(c, d) \wedge \text{link}(d, g)$
- 9 B REPLIES: $H = \text{reachable}(a, c)$
- 10 A DEDUCES: $K_3 = K_a(K_b(\text{reachable}(c, g)))$
 $K_4 = K_a(\exists i K_i(\text{reachable}(a, c)) \rightarrow K_a(\text{reachable}(a, g)))$
- 11 A : $E = \text{reachable}(a, c)$
- 12 A INDUCES: $H = \text{link}(a, c)$
-
-



Car A



Car B

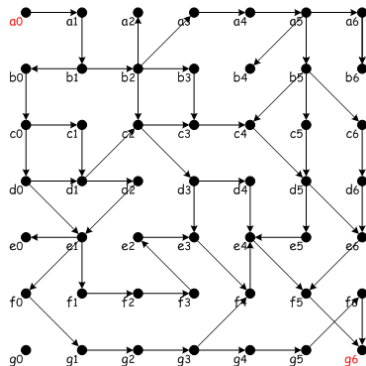


Car C

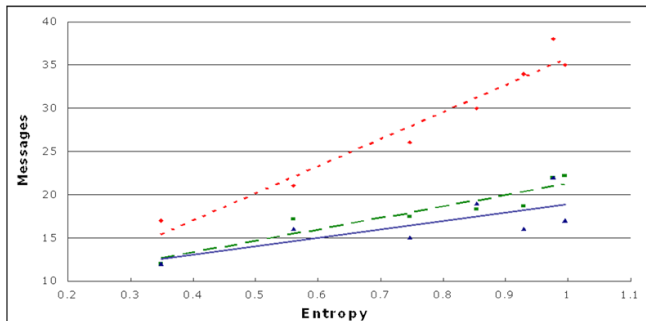
Experiments

Experiments to investigate communication cost:

- compare DILP approach against a centralized approach
- different numbers of agents, A , from 2 to 6
- varying graph sizes, G , from 60 to 120
- 100 trials for every value of A and G

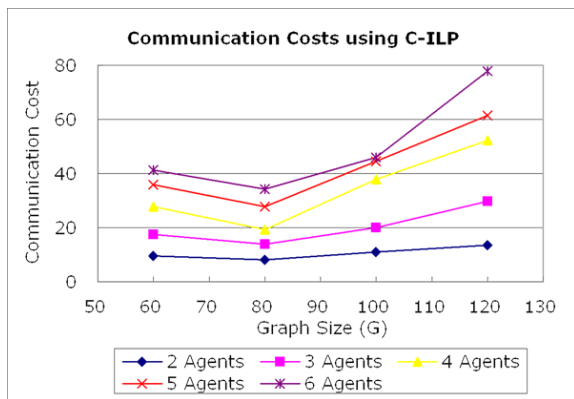


Results



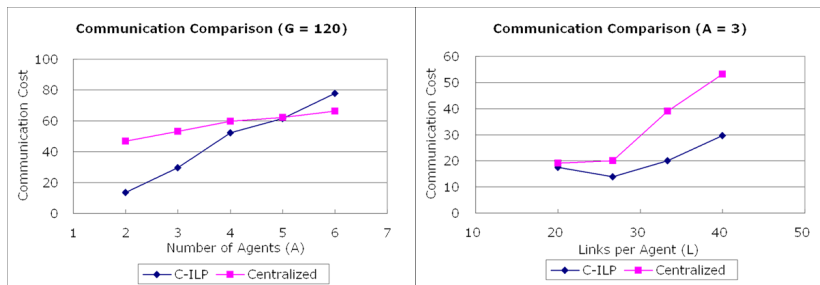
1. Communication increases as knowledge is more evenly distributed.
2. Communication increase much slower with DILP approach.

Results



3. Communication increases as number of agents increases.
4. Communication increases as graph size increases.

Results



- Communication lower with DILP in general, except when each agent knows only very few links.
- Communication lower with DILP when each agent knows 30 or more links.

Conclusion

- 1 Deduction Vs. Induction
 - Limitations with deduction and induction
 - ILP methods and techniques
- 2 Integrating Deductive and Inductive Reasoning
 - RichProlog
 - Deductive-Inductive Logic Programming Framework
 - Transformation Rules
- 3 Application in Collaborative Problem Solving
 - Collaboration in multi-agent environment
 - Integrates deduction, induction and interaction
 - Path planning example
- 4 Results show promise for:
 - overcoming the problem of knowledge being distributed;
 - avoiding central control;
 - saving communication cost.