Intention and Plan Selection for BDI Agent Systems

Sebastian Sardina

School of Computer Science and IT
RMIT University
Melbourne, Australia
sebastian.sardina@rmit.edu.au

Joint work with Lin Padgham & others...

4TH WORKSHOP ON GOAL REASONING
IJCAI-2016
July 9th, 2016
Agents@RMIT

The Agent Group is part of the Intelligent Systems area within the School of Computer Science and Information Technology. The group has basically three main areas of research: agent reasoning (e.g., goal reasoning, plan coordination, failure recovery, goal-plan conflict/resolution, etc.), agent-oriented software engineering (e.g., design, testing, software methodologies & tools, automatic code generation, etc.) and agent-based simulation (including serious games and interactive simulation, application areas focussing on climate change adaptation and environmental issues.) Besides these three main areas, staff members are involved in other related areas, such as logic and automated reasoning, computational linguistics, automated planning, machine learning, reasoning about action, etc. The group is also active within Agents-IVIC. In general, the group has interest and expertise in the following areas:

- Agent-oriented programming (mostly with BDI-type agents).
- Agent software engineering.
- Agent reasoning and reasoning about action and change.
- Computational logic and logic-based agents.
- Computational linguistics & dialogue systems.
- Automated planning.
- Agent learning.
- Agent based modelling and simulation; serious games.

We are always interested in hosting visiting researchers and periodically have postdoc positions available. Please contact Lin Padgham if you are interested in visiting our group.
### Intelligent Autonomous Behavior

<table>
<thead>
<tr>
<th></th>
<th>Behavior-based AI: set of independent simple reactive modules.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>• intelligent behavior emerges “implicitly”.</td>
</tr>
<tr>
<td></td>
<td>• popular in robotics (since the ’80).</td>
</tr>
</tbody>
</table>
Intelligent Autonomous Behavior

1. **Behavior-based AI:** set of independent simple reactive modules.
   - intelligent behavior emerges “implicitly”.
   - popular in robotics (since the ’80).

2. **Agent-oriented programming:** control specified by programmer.
   - BDI systems: JACK, JASON, 3APL, etc.
   - High-level languages: Golog-like languages, FLUX, etc.
Intelligent Autonomous Behavior

1 Behavior-based AI: set of independent simple reactive modules.
   - intelligent behavior emerges “implicitly”.
   - popular in robotics (since the ’80).

2 Agent-oriented programming: control specified by programmer.
   - BDI systems: JACK, JASON, 3APL, etc.
   - High-level languages: Golog-like languages, FLUX, etc.

3 Learning: learn how to act based on previous experience.
   - E.g., reinforcement learning.
Intelligent Autonomous Behavior

1. **Behavior-based AI:** set of independent simple reactive modules.
   - intelligent behavior emerges “implicitly”.
   - popular in robotics (since the ’80).

2. **Agent-oriented programming:** control specified by programmer.
   - BDI systems: JACK, JASON, 3APL, etc.
   - High-level languages: Golog-like languages, FLUX, etc.

3. **Learning:** learn how to act based on previous experience.
   - E.g., reinforcement learning.

4. **Automated Planning:** automatic synthesis of behavior from model.
   - **Input:** model of the world + initial state + goal to be achieved.
   - **Output:** plan or controller to achieve the goal in the world.
Intelligent Autonomous Behavior

2. **Agent-oriented programming**: control specified by programmer.
   - BDI systems: JACK, JASON, 3APL, etc.
   - High-level languages: Golog-like languages, FLUX, etc.

Here!
What are we after?

1. Understand what constitute “rational behavior” & an “agent”.
   - Theory of Practical Reasoning.
   - Informed by:
     - Philosophy of mind.
     - Psychology.
     - Computer Science.

2. Find ways to design and program agent systems
   - Informed by what rational behavior is...
   - Two areas:
     - Agent-oriented Software Engineering.
     - Agent-oriented Programming.
What are we after?

Find ways to **design** and **program** agent systems

- Informed by what rational behavior is...
- Two areas:
  - Agent-oriented Software Engineering.
  - Agent-oriented Programming.
Agent Systems

An **intelligent agent** is an **autonomous** entity, existing over time in a dynamic environment, that is able to rationally balance **pro-active** and **reactive** behavior. It perceives through **sensors** and acts through **effectors**.
Agent Systems

An intelligent agent is an autonomous entity, existing over time in a dynamic environment, that is able to rationally balance pro-active and reactive behavior. It perceives through sensors and acts through effectors.

- **autonomy**: does not require continuous external control.
- **pro-activity**: pursues goals over time; goal directed behavior.
- **reactivity**: perceives the environment and responds to it.
- **situatedness**: observe & act in the environment.
- **flexibility**: achieve goals in several ways.
- **robustness**: will try hard to achieve goals.

And also: **modular scalability** & **adaptability**!
Agent Systems

An intelligent agent is an autonomous entity, existing over time in a dynamic environment, that is able to rationally balance pro-active and reactive behavior. It perceives through sensors and acts through effectors.

- **autonomy**: does not require continuous external control.
- **pro-activity**: pursues goals over time; goal directed behavior.
- **reactivity**: perceives the environment and responds to it.
- **situatedness**: observe & act in the environment.
- **flexibility**: achieve goals in several ways.
- **robustness**: will try hard to achieve goals.

And also: **modular scalability & adaptability!**
Technology Development

Structured programming (FORTRAN, C)
Monolithic systems - Communication API (sockets)

Object Oriented programming (C++, Java, Delphi)
Client/Server - Remote Procedure Call (CORBA)

Agent Oriented Programming (BDI systems)
Distributed Control - Multi-agent frameworks (JADE)

abstraction level
distribution
complexity of domain
Multi-Agent Programming Contest

Agents in the City

Our scenario consists of two teams of agents moving through the streets of a realistic city. The goal for each team is to earn as much money as possible. Money is rewarded for completing certain jobs. Jobs comprise the acquisition, assembling, and transportation of goods. These jobs can be created by either the system (environment) or one of the agent teams. There are two kind of jobs: priced and auctioned. A team can accept an auctioned job by bidding on it. The bid amount of money is the reward. If both teams bid, naturally the lowest bid wins. If a job is not completed in time, the corresponding team is fined.

https://multiagentcontest.org/
stimulate research in multi-agent system development and programming;

identifying key problems;

collecting suitable benchmarks;

gather test cases;

test multi-agent prog. languages, platforms, tools.

We encourage submissions that specify and design a multi-agent system in terms of high-level concepts such as goals, beliefs, plans, roles, communication, coordination, negotiation, and dialogue in order to ...
International Multi-Agent Contest

- stimulate research in multi-agent system development and programming;
- identify key problems;
- collect suitable benchmarks;
- gather test cases;
- test multi-agent programming languages, platforms, tools.

We encourage submissions that specify and design a multi-agent system in terms of high-level concepts such as goals, beliefs, plans, roles, communication, coordination, negotiation, and dialogue.
International Multi-Agent Contest

• stimulate research in multi-agent system development;
• identifying key problems;
• collecting suitable benchmarks;
• gather test cases;
• test multi-agent programs, platforms, tools.

We encourage submissions that specify and design a multi-agent system in terms of high-level concepts such as goals, beliefs, plans, roles, communication, coordination, negotiation, and dialogue in order to...

Gold Mining 2005-2007
Cow Herding 2008-2010

https://multiagentcontest.org/
International Multi-Agent Contest

- stimulate research in multi-agent system development
- identifying key problems
- collecting suitable benchmarks
- gather test cases
- test multi-agent prog. languages, platforms, tools

We encourage submissions that specify and design a multi-agent system in terms of high-level concepts such as goals, beliefs, plans, roles, communication, coordination, negotiation, and dialogue in order to...

Agents on Mars

2011-2014

Sebastian Sardina, Intention and Plan Selection for BDI Agent Systems. GR’16 - July 9, 2016
BDI Programming

Learning to Select Plans

Intention Selection

Conclusions

International Multi-Agent Contest

Agents on Mars

2015-

Agents In the City

• stimulate research in multi-agent system development
• identifying key problems
• collecting suitable benchmarks
• gather test cases
• test multi-agent prog. languages, platforms, tools

We encourage submissions that specify and design a multi-agent system in terms of high-level concepts such as goals, beliefs, plans, roles, communication, coordination, negotiation, and dialogue in order to...

Gold Mining 2005-2007
Cow Herding 2008-2010
Agents on Mars 2011-2014
Agents in the City 2015-
“To ascribe beliefs, free will, intentions, consciousness, abilities, or wants to a machine is legitimate when such an ascription expresses the same information about the machine that it expresses about a person. It is useful when the ascription helps us understand the structure of the machine, its past or future behavior, or how to repair or improve it. [...] ”

John McCarthy

Question

How do we make all these ideas a concrete computational approach?
Plans and resource-bounded practical reasoning

MICHAEL E. BRATMAN
Department of Philosophy and Center for the Study of Language and Information, Stanford University, Stanford, CA 94305, U.S.A.

AND

DAVID J. ISRAEL AND MARTHA E. POLLACK
Artificial Intelligence Center and Center for the Study of Language and Information, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, U.S.A.

Received September 13, 1987
Revision accepted September 19, 1988

An architecture for a rational agent must allow for means-end reasoning, for the weighing of competing alternatives, and for interactions between these two forms of reasoning. Such an architecture must also address the problem of resource boundedness. We sketch a solution of the first problem that points the way to a solution of the second. In particular, we present a high-level specification of the practical-reasoning component of an architecture for a resource-bounded rational agent. In this architecture, a major role of the agent’s plans is to constrain the amount of further practical reasoning she must perform.

Key words: planning, practical reasoning, resource bounds.

L’architecture d’un agent rationnel doit permettre le raisonnement procédant des fins aux moyens, le choix entre différentes actions possibles, et l’interaction entre ces deux modes de raisonnement. Elle doit aussi tenir compte des conséquences des limites de ressources disponibles. Nous esquissons ici une solution au premier problème qui indique comment on pourrait résoudre le second. Nous proposons, en particulier, une spécification abstraite d’un module de génération de plans pour un agent rationnel dont les ressources sont bornées. Dans cette architecture, le rôle principal des plans d’un agent est de limiter les ressources devant être consacrés au raisonnement.

Mots clés : planification, raisonnement pratique, limites de ressources.


IRMA Architecture
(Intelligent Resource-bounded Machine Architecture)
IRMA Architecture [Bratman, Israel, Pollack CI’88]

Fig. 1. An architecture for resource-bounded agents.
Detailed BDI Architecture

The BDI architecture consists of several key components:

- **SENSORS** receive information about the world.
- **ACTUATORS** execute actions that achieve the agent's intentions.
- **Environment** provides the context for the agent's actions.
- **Beliefs** store the agent's knowledge and beliefs about the world.
- **Intention Stacks** consist of multiple stacks, each containing pending events that the agent needs to resolve.
- **Plan library** contains recipes for handling goals and events.
- **BDI engine** processes events, updates beliefs, and selects actions.
- **Pending Events** are events that the agent needs to address.

Rational behavior arises due to the agent committing to some of its desires and selecting actions that achieve its intentions given its beliefs.
**Detailed BDI Architecture**

- **SENSORS**
- **ACTUATORS**

**Environment**

- **Beliefs**
- **Pending Events**
- **Intention Stacks**
- **BDI engine**
- **Plan library**
  - $e_1 : \psi_1 \leftarrow \delta_1$
  - ...
  - $e_n : \psi_n \leftarrow \delta_n$

**goals/desires to resolve**

- **events**

Rational behavior arises due to the agent committing to some of its desires, and selecting actions that achieve its intentions given its beliefs.
**Detailed BDI Architecture**

- **SENSORS**:
  - Information about the world

- **ACTUATORS**:
  - Actions

- **Environment**

- **Pending Events**
  - Goals/desires to resolve

- **Beliefs**

- **Intention Stacks**

- **BDI engine**

- **Plan library**

- **Actions**

Rational behavior arises due to the agent committing to some of its desires, and selecting actions that achieve its intentions given its beliefs.
Detailed BDI Architecture

**SENSORS**
- Environment
- Information about the world

**ACTUATORS**

**Pending Events**
- Goals/desires to resolve
- Recipes for handling goals-events

**Beliefs**
- Intention Stacks
- BDI engine
- Plan library

**BDI engine**
- Actions

**Plan library**
- $e_1 : \psi_1 \leftarrow \delta_1$
- ...
- $e_n : \psi_n \leftarrow \delta_n$

**Intention Selection**

Rational behavior arises due to the agent committing to some of its desires, and selecting actions that achieve its intentions given its beliefs.
Rational behavior arises due to the agent committing to some of its desires, and selecting actions that achieve its intentions given its beliefs.
Detailed BDI Architecture

### Environment
- **SENSEORS**
  - Information about the world
- **ACTUATORS**
  - Partially uninstantiated programs with commitment

### Parts of BDI Logic
- **Beliefs**
  - Recipes for handling goals-events
- **Pending Events**
- **Intention Stacks**
- **Plan Library**
  - $e_1 : \psi_1 \leftarrow \delta_1$
  - $\vdots$
  - $e_n : \psi_n \leftarrow \delta_n$

### Rational Behavior
- Arises due to the agent committing to some of its desires and selecting actions that achieve its intentions given its beliefs.

Rational behavior arises due to the agent committing to some of its desires, and selecting actions that achieve its intentions given its beliefs.
The BDI Execution Cycle [Rao&Georgeff 92]
The BDI Execution Cycle [Rao&Georgeff 92]

- **Event**
  - $e$

- **Beliefs**
  - Query $\psi$

- **Select plan based on situation**

- **Plan library**
  - $e_1 : \psi_1 \leftarrow \delta_1$
  - $\vdots$
  - $e_n : \psi_n \leftarrow \delta_n$
The BDI Execution Cycle [Rao&Georgeff 92]

- Event
- Beliefs
- Select plan based on situation
- Plan library
  \[ e_1 : \psi_1 \leftarrow \delta_1 \]
  \[ \vdots \]
  \[ e_n : \psi_n \leftarrow \delta_n \]

- Current Intentions
- Query \( \psi \)
- Push \( \delta \)
The BDI Execution Cycle [Rao&Georgeff 92]

Event $e$ → Beliefs

query $\psi$

Select plan based on situation

$e_1 : \psi_1 \leftarrow \delta_1$

$\vdots$

$e_n : \psi_n \leftarrow \delta_n$

Action

Sebastian Sardina, Intention and Plan Selection for BDI Agent Systems. GR’16 - July 9, 2016
The BDI Execution Cycle [Rao&Georgeff 92]

- Event $e$ queries beliefs $\psi$
- Select plan based on situation
- Plan library
  - $e_1 : \psi_1 \leftarrow \delta_1$
  - $\vdots$
  - $e_n : \psi_n \leftarrow \delta_n$
- Push $\delta$
- Action
- Step on some intention
- Current Intentions
The BDI Execution Cycle [Rao&Georgeff 92]

1. **Event**: e
2. **Beliefs**: \( \psi \)
3. **Select plan based on situation**
   - \( e_1 : \psi_1 \leftarrow \delta_1 \)
   - \( \vdots \)
   - \( e_n : \psi_n \leftarrow \delta_n \)
4. **Plan library**
5. **Current Intentions**
6. **Action**: step on some intention
7. **Update**
8. **Sub-goal generation**
9. **Query** \( \psi \)
10. **Push** \( \delta \)
Some BDI Agent-oriented Programming Languages

Some BDI programming language systems/platforms/architectures:

1. PRS & dMars
2. 3APL, http://www.cs.uu.nl/3apl/
3. GOAL, http://ii.tudelft.nl/trac/goal/
4. 2APL, http://apapl.sourceforge.net/
Defining Agents in JACK: Player.agent

Base class: aos.jack.jak.agent.Agent

```java
public agent Player extends Agent {
    #has capability ClimaTalking cap;
    #handles event PerceiveClimaServer;
    #handles event EExecuteCLIMAAction;
    #handles event EAct;
    #posts event EExecuteCLIMAAction ev_executeAction;
    #sends event EInformLoc ev_informLoc;
    ... 
    #uses plan MoveRandomly;
    #uses plan PickGold;
    #uses plan HandlePercept;
    ...
    #private data GoldAt bel_goldAt();
    #private data CurrentPosition bel_currPosition();
    #private data NumCarryingGold bel_noCarrGold();
    ... 
}
```
Prometheus Design Tool (PDT)

1. **Design:** of an agent system in 3 interrelated phases.
2. **Code generation:** skeleton code in JACK agent language.

Possibility of Many Options

**BDI execution** = delayed context subgoal expansion + failure recovery
Possibility of Many Options

BDI execution = delayed context subgoal expansion + failure recovery

subgoals

Goal
different ways to achieve goal
Possibility of Many Options

BDI execution = delayed context subgoal expansion + failure recovery

BDI Programming
= Implicit Goal-based Programming
+ Rational Online Executor
From David’s talk: *Better future performance*

1. Avoid dead-ends with respect to current goals.

2. Avoid states to jeopardize goal achievement in future.

3. Take actions to maximize actions and goal in the future.

… or something on these lines. :-(
Two Core Deliberation Tasks

Standard Rational Executor/Reasoner
[Rao and Georgeff 1992, Bratman et al. 1988]

1 select pending event-goals to handle (deliberation and filtering).
2 select a plan to handle goal & commit to it (means-end reas.).
3 select intention and execute part of it (execution).
Two Core Deliberation Tasks

Standard Rational Executor/Reasoner
[Rao and Georgeff 1992, Bratman et al. 1988]

2. select a plan to handle goal & commit to it (*means-end reas.*).

3. select intention and execute part of it (*execution*).
Two Core Deliberation Tasks

Standard Rational Executor/Reasoner
[Rao and Georgeff 1992, Bratman et al. 1988]

2. select a plan to handle goal & commit to it (*means-end reas.*).
3. select intention and execute part of it (*execution*).

Standard approaches:

- Let the BDI user *program* both selections
  - meta-reasoning plans, deliberation cycle programming, preferences.
- Select from several *built-in schemes*.
  - random, top-down, round-robin, FIFO.
- Select based on *additional domain information*.
  - priorities, deadlines, reward and cost, etc.
Towards Better Plan & Intention Selection

Smarter plan & intention selection under 3 constraints:

- **Domain-independent**: no extra domain information required.
- **No major overhead** on BDI executor.
- **Easily incorporated** into existing BDI platforms.

General approach

- **Plan Selection**: Prefer plans that have succeeded in similar situations.
  - Learn/improve plans' context conditions.
  - Induce decision trees for plans' based on executions.
  - Rely on **WEKA** package.

- **Intention Selection**: Prefer most “vulnerable” applicable intention.
  - How much **know-how** is available for a goal-event?
  - Reason on plan/goal coverage using model counting.
Towards Better Plan & Intention Selection

Smarter plan & intention selection under 3 constraints:
- **Domain-independent**: no extra domain information required.
- **No major overhead** on BDI executor.
- **Easily incorporated** into existing BDI platforms.

General approach

**Plan Selection**: Prefer plans that have succeeded in similar situations.
- Learn/improve plans’ context conditions.
- Induce decision trees for plans’ based on executions.
- Rely on WEKA package.
- [AAMAS’10, JRAS’10, IJCAI’11]
Towards Better Plan & Intention Selection

Smarter plan & intention selection under 3 constraints:
- **Domain-independent**: no extra domain information required.
- No major overhead on BDI executor.
- Easily incorporated into existing BDI platforms.

General approach

**Plan Selection:** *Prefer plans that have succeeded in similar situations.*
- Learn/improve plans’ context conditions.
- Induce decision trees for plans’ based on executions.
- Rely on **WEKA** package.
- [AAMAS’10, JRAS’10, IJCAI’11]

**Intention Selection:** *Prefer most “vulnerable” applicable intention.*
- How much *know-how* is available for a goal-event?
- Reason on plan/goal coverage using **model counting**.
- [AAMAS’12, AAMAS’14, JAAMAS’15]
Towards Better Plan & Intention Selection

Smarter plan & intention selection under 3 constraints:

- **Domain-independent**: no extra domain information required.
- **No major overhead** on BDI executor.
- **Easily incorporated** into existing BDI platforms.

**General approach**

**Plan Selection**: *Prefer plans that have succeeded in similar situations.*
- Learn/improve plans’ context conditions.
- Induce **decision trees** for plans’ based on executions.
- Rely on **WEKA** package.
- [AAMAS’10, JRAS’10, IJCAI’11]
The BDI Execution Cycle [Rao&Georgeff 92]

Event → query $\psi$ → Select plan based on situation → push $\delta$ → Current Intentions

Plan library:

$e_1 : \psi_1 \leftarrow \delta_1$
$\vdots$
$e_n : \psi_n \leftarrow \delta_n$
Recap. Plan Selection

- **Problem:** Context conditions hard to craft & environment changes.
Recap. Plan Selection

- **Problem**: Context conditions hard to craft & environment changes.
- **Approach**: Attach decision tress to plans + confidence measure.
  - Easy to incorporate by using learning packages (e.g., WEKA)

Plan Selection relative to *confidence level*:

- Plan success rate (as per DT).
- Plan Stability.
- World novelty rate.
Recap. Plan Selection

- **Problem:** Context conditions hard to craft & environment changes.
- **Approach:** Attach decision tree to plans + confidence measure.
  - Easy to incorporate by using learning packages (e.g., WEKA)

Plan Selection relative to confidence level:

- Plan success rate (as per DT).
- Plan Stability.
- World novelty rate.

- **Empirical Evaluation:** agent learns to succeed & adapts
  - Synthetic programs of various shapes [AAMAS’10]
  - Hanoi Tower [JRAS’10]
  - Battery Controller [IJCAI’11]
Given net building demand, calculate an appropriate battery response in order to maintain grid power consumption within range $[0, \rho_h]$. 
Recovery from **temporary module failures** during $[0, 20k]$, $[20k, 40k]$ episodes.
Towards Better Plan & Intention Selection

Smarter plan & intention selection under 3 constraints:

- **Domain-independent**: no extra domain information required.
- No major overhead on BDI executor.
- Easily incorporated into existing BDI platforms.

General approach

**Plan Selection:** Prefer plans that have succeeded in similar situations.

- Learn/improve plans’ context conditions.
- Induce decision trees for plans based on executions.
- Rely on **WEKA** package.
  
- [AAMAS’10, JRAS’10, IJCAI’11]

**Intention Selection:** Prefer most “vulnerable” applicable intention.

- How much **know-how** is available for a goal-event?
- Reason on plan/goal coverage using **model counting**.

- [AAMAS’12, AAMAS’14, JAAMAS’15]
The BDI Execution Cycle [Rao&Georgeff 92]

Event \( e \) → Beliefs \( \psi \) → Select plan based on situation → Plan library

\[ e_1 : \psi_1 \leftarrow \delta_1 \]
\[ \vdots \]
\[ e_n : \psi_n \leftarrow \delta_n \]

Step on some intention

Current Intentions

Sebastian Sardina, Intention and Plan Selection for BDI Agent Systems. GR’16 - July 9, 2016
Intention Selection
How to choose which intention to progress next?

**Issues** intention interference + dynamic environment + incomplete know-how

**Objective** maximize successfully executed intentions
Intention Selection

How to choose which intention to progress next?

**Issues** intention interference + dynamic environment + incomplete know-how

**Objective** maximize successfully executed intentions

**Standard approaches:**

**Simple** first-in-first-out (FIFO) and round-robin (RR)

**Meta-level programming** deliberation cycle, call-back hooks, etc.

**Domain info** priorities, deadlines, value, dependencies, etc.
Intention Selection

How to choose which intention to progress next?

**Issues** intention interference + dynamic environment + incomplete know-how

**Objective** maximize successfully executed intentions

**Standard approaches:**

**Simple** first-in-first-out (FIFO) and round-robin (RR)

**Meta-level programming** deliberation cycle, call-back hooks, etc.

**Domain info** priorities, deadlines, value, dependencies, etc.

**Challenge:** intelligent, domain-independent intention selection

- improves intention success;
- improves focus of attention;
- low over-head;
- easy to implement.
Low Coverage Prioritization

Idea: Opportunistically execute the most “vulnerable” intention

Intentions contain unresolved goals...

How much “know-how” is available?

Less know how, more vulnerable...

\[
P_1 \quad G_2 \quad G_3 \quad G_4
\]

\[
P_3 \quad P_4 \quad P_5 \quad P_6 \quad P_7 \quad P_8
\]
Low Coverage Prioritization [AAMAS’12]

Idea: Opportunistically execute the most “vulnerable” intention

Intentions contain unresolved goals...
How much “know-how” is available?
Less know how, more vulnerable...

Plan coverage = % states applicable
Goal coverage = % states with app. plans
Aggregated coverage = considers know-how below hierarchy
Lower the coverage, more vulnerable intention

\[
\begin{align*}
G_1 & \quad P_1 \quad G_1 \\
G_2 & \quad P_3 \quad G_2 \\
G_3 & \quad P_4 \quad P_5 \\
G_4 & \quad P_6 \quad P_7 \quad P_8 \\
p & \quad \neg p \\
q & \quad r \lor s \\
s \lor t \\
s \land t \\
s \land \neg t \\
\neg s \land t
\end{align*}
\]
Low Coverage Prioritization

Idea: Opportunistically execute the most "vulnerable" intention

Intentions contain unresolved goals...
How much "know-how" is available?
Less know how, more vulnerable...

Plan coverage = % states applicable
Goal coverage = % states with app. plans
Aggregated coverage = considers
know-how below hierarchy
Lower the coverage, more vulnerable intention

Pick intention with lowest-coverage!
Experimental Results: Impact on Intention Success

![Graphs showing the impact of different strategies on intention success]

- Improves success consistently!
Experimental Results: Impact on Intention Success

(a) $C - \text{FIFO}$

(b) $C - \text{RR}$

Impossible to beat!

Improves success consistently!
Coverage of Plans and Goals

Coverage of a plan = % of states where plan is applicable
Coverage of a goal = % of states with plans available
Overlap of plans = plans applicable simultaneously
Aggregated coverage = know-how below hierarchy + overlap

\[ 0.59375 = 0.5 \times 0.4375 + 0.5 \times 0.75 \]
Enablement Contribution

Major component is goal enablement checking!

(c) $C \rightarrow \text{FIFO}^E$

(d) $C \rightarrow \text{RR}^E$
Enablement Addition to FIFO and RR on Hanoi

Intention success rate

Efficiency (optimal = 31 moves)
Enablement Addition to FIFO and RR on Hanoi II

![Graphs showing Failure recovery rate and Fairness vs Engine deterioration rate](image)

- **Failure recovery rate**
  - FIFO
  - FIFO<sup>E</sup>
  - RR
  - RR<sup>E</sup>

- **Fairness**
  - Engine deterioration rate
Summary

- Intention & Plan selection at the core of BDI “intelligence”
  - ... but almost not addressed (in domain-independent way)!
• Intention & Plan selection at the core of BDI “intelligence”
  • ... but almost not addressed (in domain-independent way)!

• Proposed learning-based plan selection:
  • Attach a decision tree as a “learnt” context-condition.
  • Use of confidence measure to balance exploit/explore.
  • **Experimental Results:**
    • Converges to optimal
    • Adapts to changes.

• Proposed low-coverage prioritization:
  • Pick most “know-how vulnerable” intention: via coverage
  • **Experimental Results:**
    • Increases the success rate (almost always).
    • Better in low-coverage + highly dynamic situations.
    • Improves RR significantly with little loss on fairness.
    • Improves efficiency & decreases failure recovery.

Both approaches domain-independent & implementable.
Summary

• Intention & Plan selection at the core of BDI “intelligence”
  • ... but almost not addressed (in domain-independent way)!

• Proposed learning-based plan selection:
  • Attach a decision tree as a “learnt” context-condition.
  • Use of confidence measure to balance exploit/explore.
  • Experimental Results:
    • Converges to optimal
    • Adapts to changes.

• Proposed low-coverage prioritization:
  • Pick most “know-how vulnerable” intention: via coverage
  • Experimental Results:
    • Increases the success rate (almost always).
    • Better in low-coverage + highly dynamic situations.
    • Improves RR significantly with little loss on fairness.
    • Improves efficiency & decreases failure recovery.
Summary

• Intention & Plan selection at the core of BDI “intelligence”
  • ... but almost not addressed (in domain-independent way)!

• Proposed learning-based plan selection:
  • Attach a decision tree as a “learnt” context-condition.
  • Use of confidence measure to balance exploit/explore.
  • Experimental Results:
    • Converges to optimal
    • Adapts to changes.

• Proposed low-coverage prioritization:
  • Pick most “know-how vulnerable” intention: via coverage
  • Experimental Results:
    • Increases the success rate (almost always).
    • Better in low-coverage + highly dynamic situations.
    • Improves RR significantly with little loss on fairness.
    • Improves efficiency & decreases failure recovery.

• Both approaches domain-independent & implementable.
Challenges?

Need more advanced **built-in infrastructure support for goal reasoning**:

1. domain-independent;
2. automatic;
3. not based on new programming constructs;
4. based on knowledge representation & learning!
Challenges?

Need more advanced built-in infrastructure support for goal reasoning:

1. domain-independent;
2. automatic;
3. not based on new programming constructs;
4. based on knowledge representation & learning!

Promising challenges:

1. Plan & intention selection: *key places of deliberation*!
2. Goals and plan integration: *representation & reasoning*
4. Goal generation/creation: *basic motivations/desires?*
5. Verification & debugging techniques/tools.
Thank you for your attention!

... and thanks to those who contributed to this work:

Lin Padgham

John Tangarajah

Dhirendra Singh

Max Waters
References: Agents

Michael E. Bratman, David J. Israel, and Martha E. Pollack.
Plans and resource-bounded practical reasoning.

An abstract architecture for rational agents.

Agentspeak(L): BDI agents speak out in a logical computable language.

Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge.
*Programming Multi-agent Systems in AgentSpeak Using Jason.*


References: Intention Selection

John Thangarajah, Sebastian Sardina, and Lin Padgham.
Measuring plan coverage and overlap for agent reasoning.

Max Waters, Lin Padgham, and Sebastian Sardina.
Evaluating coverage based intention selection.
Nominated for Jodi Best Student Paper award.

Max Waters, Lin Padgham, and Sebastian Sardina.
Improving domain-independent intention selection in BDI systems.

Raj Jain, Dah-Ming Chiu, and William R Hawe.
*A quantitative measure of fairness and discrimination for resource allocation in shared computer system.*
A Dynamic Confidence Measure

Observe & record on averaging window $n$:

- rate of plan success.
- plan local stability: success rate $> \epsilon$?
- plan global stability: ratio of stable plans below in the goal-tree.
- rate of new worlds seen.

Confidence Measure for Plans

Stability measure $C_s(P, w, n)$:
- how well-informed the last $n$ executions of plan $P$ in world $w$ were?

World metric $C_w(P, n)$:
- how much we have seen the "interesting" worlds for $P$?

Aggregated confidence measure $C(P, w, n)$:
- $C(P, w, n) = \alpha C_s(P, w, n) + (1 - \alpha) C_w(P, n)$
A Dynamic Confidence Measure

Observe & record on averaging window $n$:
- rate of plan success.
- plan local stability: success rate $> \epsilon$?
- plan global stability: ratio of stable plans below in the goal-tree.
- rate of new worlds seen.

Confidence Measure for Plans

- Stability measure $C_s(P, w, n)$:
  - how well-informed the last $n$ executions of plan $P$ in world $w$ were?
- World metric $C_w(P, n)$:
  - how much we have seen the “interesting” worlds for $P$?
- Aggregated confidence measure $C(P, w, n)$:
  - $C(P, w, n) = \alpha C_s(P, w, n) + (1 - \alpha) C_w(P, n)$
Example: Dynamic Confidence Measure

After $E=15$, $P_c$ starts to fail. The confidence drops, promoting new exploration and re-learning.
Plan Selection via Plan Weighting

Given $P$'s confidence measure $C(P, w, n)$ & DT estimation $\mathcal{P}(P, w)$:

**Plan Weight**

Using predicted likelihood of success & confidence measure:

$$\Omega(P, w, n) = 0.5 + [C(P, w, n) \times (\mathcal{P}(P, w) - 0.5)]$$

- When $C(P, w, n) = 1$, then $\Omega(P, w, n) = \mathcal{P}(P, w)$
- When $C(P, w, n) = 0$, then $\Omega(P, w, n) = .5$ (default weight)

**BDI Plan selection:** probabilistically+proportionally to plans’ weights.