IMPACTing SHOP:

Planning in a Multi-Agent Environment

(From SHOP via A-SHOP to shop)

J. Dix, H. Munoz-Avila, Dana Nau

University of Koblenz, University of Maryland

Overview

1. HTN Planning with SHOP
2. IMPACT
3. A-SHOP
4. Implementation: shop
5. Conclusions

1 HTN-Planning

Decompose tasks into subtasks until you **end up with primitive tasks** that can be solved by operators.

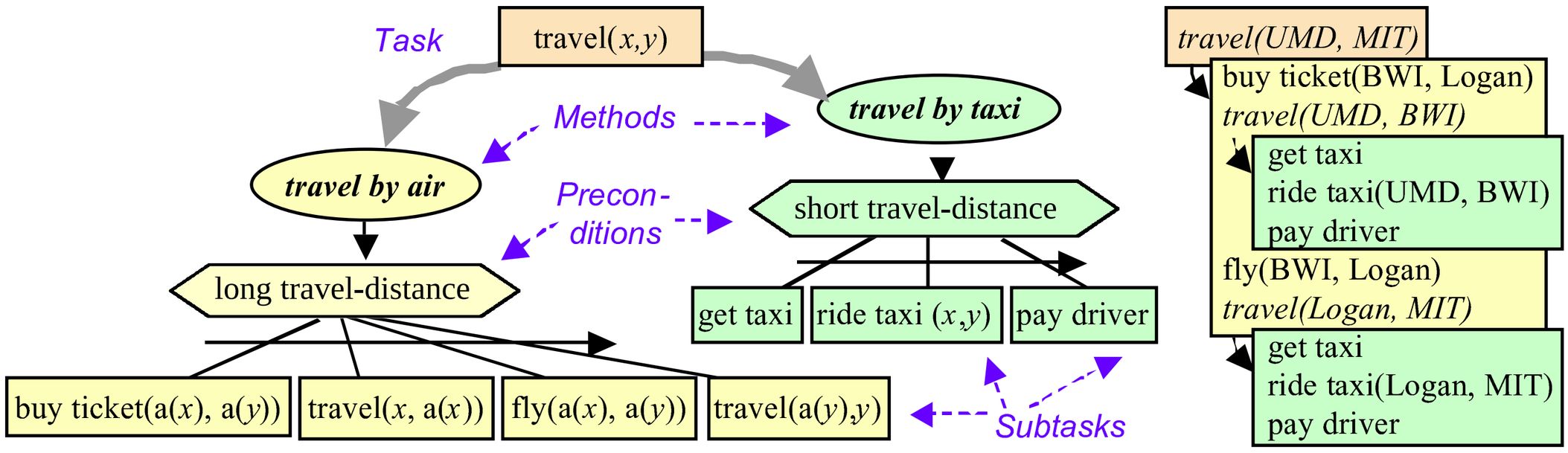
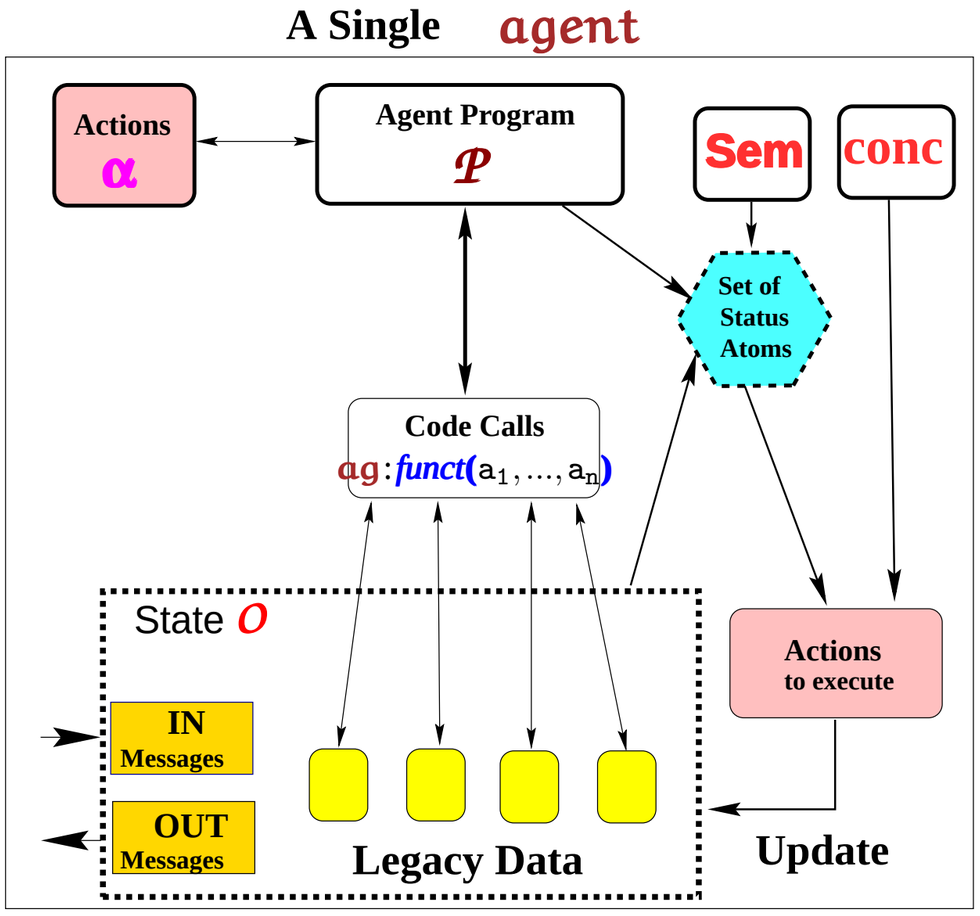


Figure 1.1: Travel planning example.

2 IMPACT

Comparing IMPACT and SHOP

IMPACT	SHOP
heterogenous data	logic
distributed data	main memory
no planning built-in	**efficient domain independent planner**
reasoning with arbitrary legacy code	only symbolic reasoning



Code Call: returns a set

$\texttt{agent}:\textit{function}(\texttt{arg}_1,\ldots,\texttt{arg}_n)$

Code Call Atoms: boolean

$\textbf{in}(\texttt{X},\texttt{agent}:\textit{function}(\texttt{arg}_1,\ldots,\texttt{arg}_n))$

Code Call Condition (ccc):

Conjunction of code call atoms,

$\textbf{in}(\texttt{X},\texttt{ag}:f(\texttt{Y},\texttt{c}))\wedge\textbf{in}(\texttt{Y},\texttt{ag}':g(\texttt{d}))$

Action Status Atoms:

Op α,where Op $\in\{\textbf{Do},\textbf{P},\textbf{F},\textbf{O}\}$

Agent Program: Rules like

$\text{Op}\alpha\leftarrow$

$\text{Op}_1\beta_1,\ldots,\text{Op}_n\beta_n,\textbf{ccc}_1,\ldots,\textbf{ccc}_m$

3 A-SHOP

How can we incorporate SHOP into IMPACT?

Methods: **Decompose** tasks into subtasks, if preconditions are met.

Operators: Solve **primitive** tasks by a executing operators.
They modify the local knowledge base.

SHOP's operators are close to **IMPACT's actions.**

- **SHOP's operators** change the internal state.
- **IMPACT's actions** modify the agent's state O.

Definition 3.1 (Agentized Method)
An **agentized method** *is an expression of the form (*: **AgentMeth h χ t***) where*

- **h** *(the method's* head*) is a compound task,*
- **χ** *(the method's* preconditions*) is a code call condition, and*
- **t** *is a task list.*

Definition 3.2 (Agentized Operator)
An **agentized operator** *is an expression of the form* **(**: **AgentOp h** χ_{add} χ_{del}**)**, *where*

- **h** *(the* head*) is a primitive task and,*
- χ_{add} *and* χ_{del} *are code call conditions (called the* add- *and* delete-lists*).*

The set of variables in χ_{add} *and* χ_{del} *is a subset of the set of variables in* **h**.

Head:

AirTransport(LocFrom, LocTo, Cargo, CargoWeight)

Preconditions:

in(CargoPL, **supplier**:*cargoPlane*(LocFrom))&
in(Dist, **statistics**:*distance*(LocFrom, LocTo))&
in(DCargoPL, **statistics**:*authorRange*(CargoPL))&
Dist $\leq$ DCargoPL&
in(CCargoPL, **statistics**:*authorCapacity*(CargoPL))&
CargoWeight $\leq$ CCargoPL&

Subtasks:

load(Cargo, LocFrom)
fly(Cargo, LocFrom, LocTo)
unload(Cargo, LocTo)

Figure 3.1: Agentized method for a military logistics problem.

procedure *A-SHOP*$(\mathbf{t}, \mathcal{D})$

1. **if t** $=$ *nil* **then return** *nil*
2. $t :=$ the first task in **t**; $R :=$ the remaining tasks
3. **if** t is primitive and a simple plan for t exists **then**
4. $\quad q :=$ **simplePlan**(t)
5. $\quad$ **return** $cons(q, \text{A-SHOP}(R, \mathcal{D}))$
6. **else if** t is non-prim. $\wedge$ $\exists$ a reduction of t **then**
7. $\quad$ **nondeterministically** choose a reduction:
   Nondeterministically choose an agentized method,
   (: **AgentMeth** $h\chi t^{red}$**)**, with μ the most general unifier of h and t and substitution θ in
   IMPACT : $instances(\chi)$.
8. $\quad$ **return** $\textit{A-SHOP}(append(t^{red}\mu\theta, R), \mathcal{D})$
9. **else return** *FAIL*
10. **end if**

end *A-SHOP*

procedure **simplePlan**(t)

11. **nondeterministically** choose agentized operator
    Op $=$ **(**: **AgentOp** $h\chi_{add}\chi_{del}$**)** with ν the most general unifier of h and t s.t. h is ground
12. IMPACT : $apply(\text{Op}\nu)$
13. **return** Opν

end *A-SHOP*

Figure 3.2: A-SHOP, the agentized version of SHOP.

safe: Ensures that when code calls are invoked, all arguments are ground.

strongly safe: Ensures that when code calls are invoked, they terminate.

strongly safeness is a compile time check!

Theorem 3.1 (Sound-, Completeness)
Let $\mathcal{O}$ be a state and $\mathcal{D}$ be a collection of agentized methods and operators.
If **all** *the* **preconditions** *in the agentized methods and* **add-** *and* **delete-lists** *in the agentized operators are* **strongly safe** *w.r.t. the variables in the method (resp. operator), then A-SHOP is correct and complete.*

4 Implementation

- SHOP is implemented in LISP: IJCAI '99: Nau, Cao, Lotem, Munoz-Avila , Detailed TR, Feb. 2000.
- IMPACT is running on a Windows platform, implemented in JAVA. CCC's can be evaluated over wide variety of data structures and software packages:

  Heterogenous Active Agents, MIT Press, 2000.
  Subrahmanian, Bonatti, Dix, Eiter, Kraus, Ozcan, Ross. MIT Press, 2000

  AIJ articles (1999, 2000) of Eiter/Subrahmanian/Pick. Various Extensions wrt. beliefs, time, probabilistic reasoning, security in AIJ, JLP, TOCL.

- A-SHOP rebuilt in JAVA, with communication module to execute ccc's.
  shop is on its way.

Military Logistics Planning

Information about different assets is heterogenous and distributed (various DBMS's).

```
statistics:distance(Loc1,Loc2)
statistics:authorRange(Aircraft)
statistics:authorCapacity(Aircraft)
supplier:cargoPlane(Loc)
```

Figure 4.1: Code-calls in the military logistics domain.

5 Conclusion

- A-SHOP, modified version of SHOP.
  Leads to **shop**: a **planning agent** in IMPACT.
- A-SHOP plans with heterogeneous, distributed information sources.
  Handling resources separately from the planning can improve performance.

- **shop** will interact with other agents.
- Interleave planning with plan execution.
- Coordination of multiple copies of **shop**.