

For a given planner-domain pair

- How to improve performance? *Assist with knowledge.*
- How to represent knowledge? *Use macro-actions.*
- What macro-actions improve performance? *Wizard suggests.*

Related Work

Characteristic	Domain/Planner	Solution
Plateau escaping sequence	Fast Forward	MARVIN
Valley travelling sequence	One heuristic planner	MACLEARN
Causally linked macro	Fast Forward	Macro-FF
<i>Don't care; don't know</i>	<i>Any planner</i>	<i>Wizard</i>
Symmetry, almost symmetry	Briefcase, Gripper, etc.	MARVIN
Component abstraction	Gripper, Rover, etc.	Macro-FF
Operator decomposability	Rubik's cube, etc.	Korf's MPS
<i>Don't care; don't know</i>	<i>Any domain</i>	<i>Wizard</i>

Wizard: Suggesting Macro-Actions Comprehensively

Hakim Newton, John Levine

Strathclyde Planning Group
Computer and Information Sciences
University of Strathclyde
Glasgow, United Kingdom

Doctoral Consortium, ICAPS'2007

Macro-Actions or Macros

Broader Perspective

- Groups or sequences of actions or macro-actions
- Like compound statements or procedures in programming
- Useful for conceptualisation and performance improvement
- Should readily be available and widely applicable

Planning Perspective

- Represent conceptually and/or technically useful subplans
- Not a formally accepted standard concept in PDDL
- Only standard way to reason: macros in disguise of actions
- Can only provide additional choices, not obligations

Scope of Wizard

Acquisition

Representation

Exploitation

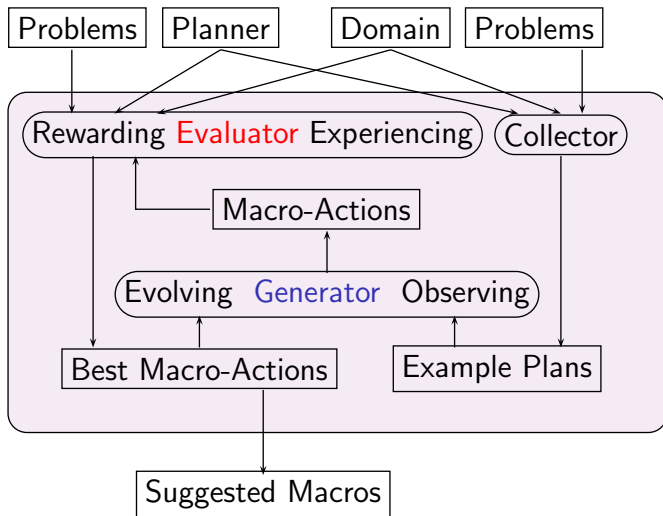
Macro representation and exploitation: **nothing done here**

- We assume these two are given or can be obtained somehow.
- PDDL does not support macros and planners are not aware of.
- We compile macros into actions in STRIPS and FLUENTS.
- We hope macros will be formally recognised in planning.

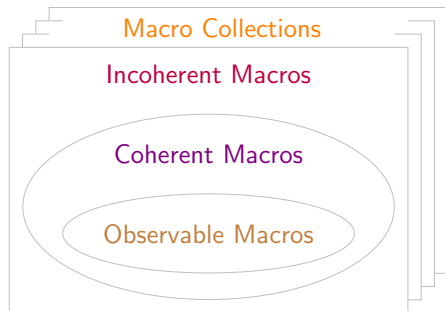
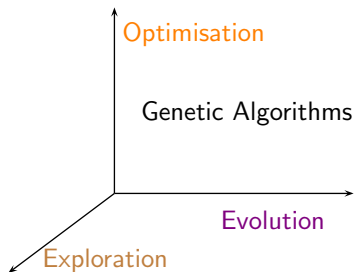
Macro acquisition: **main objective**

- Learning macros for **arbitrary** planners and domains
- Arbitrariness is in **structural properties**, not just PDDL levels
- Improving performance making **no modification** to a planner

System Architecture: Control/Data Flow Diagram



Genetic Algorithms on Macros



Evolution of Macros

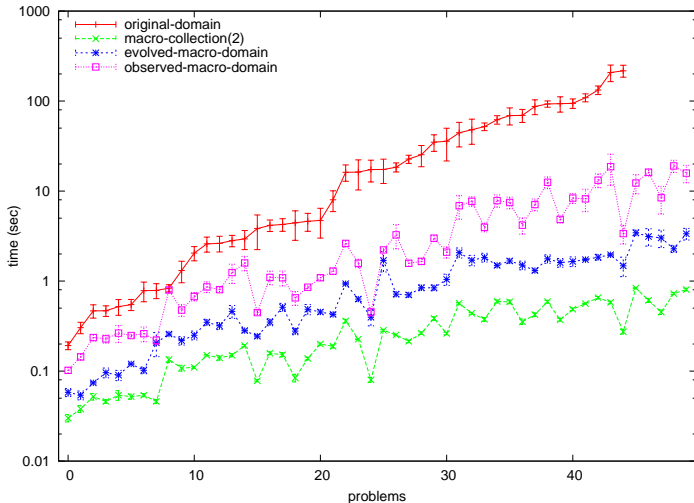
Non-observable and Incoherent Macros: Surprise?

- Given examples do not always cover all systems aspects.
- Planners often prune many useful choices.
- Relaxed domains are incoherent but useful.
- Incoherent macros may be useful in the same way.

Blocks-FF

<i>Was Observable</i>	<i>Was non-observable</i>	<i>Incoherent</i> <i>(drop ?b5)</i>
(unstack ?b5 ?b0)	(unstack ?b5 ?b0)	(unstack ?b5 ?b0)
(stack ?b5 ?b2)	(stack ?b5 ?b2)	(stack ?b5 ?b2)
(pick ?b0)	(pick ?b0)	(pick ?b0)
	(stack ?b0 ?b5)	(stack ?b0 ?b5)

Blocks-LPG



Learning macros for

- The state-of-the-art planners having different base architectures
- Benchmark domains having diverse structural features

Learning macros of type

- Causally linked, coherent, incoherent, intuitively natural
- Attaining interacting subgoals, plateau escaping
- Plan time improving, plan length improving
- Domain specific, planner specific
- Observable, non-observable
- Reagent, Catalytic
- Chunks, bunches

Conclusion

