

# How to Achieve Faster Planning?

When a planner and a domain are given

- ▶ Improve the planner by tackling its drawbacks
- ▶ Get a domain encoding that suits the planner
- ▶ Acquire knowledge and exploit during planning

What means?

- ▶ Macro-actions
- ▶ Control-rules
- ▶ Control-policies.

What Macro-Actions?

- ▶ How to acquire?
- ▶ How to exploit?
- ▶ How to deliver?

**How to learn macro-actions in a given context?**  
How to make macro-actions readily available in planning?

# How to Achieve Faster Planning?

When a planner and a domain are given

- ▶ Improve the planner by tackling its drawbacks
- ▶ Get a domain encoding that suits the planner
- ▶ Acquire knowledge and exploit during planning

What means?

- ▶ Macro-actions
- ▶ Control-rules
- ▶ Control-policies.

What Macro-Actions?

- ▶ How to acquire?
- ▶ How to exploit?
- ▶ How to deliver?

**How to learn macro-actions in a given context?**

How to make macro-actions readily available in planning?

# How to Achieve Faster Planning?

## When a planner and a domain are given

- ▶ Improve the planner by tackling its drawbacks
- ▶ Get a domain encoding that suits the planner
- ▶ Acquire knowledge and exploit during planning

## What means?

- ▶ Macro-actions
- ▶ Control-rules
- ▶ Control-policies.

## What Macro-Actions?

- ▶ How to acquire?
- ▶ How to exploit?
- ▶ How to deliver?

How to learn macro-actions in a given context?

How to make macro-actions readily available in planning?

# How to Achieve Faster Planning?

## When a planner and a domain are given

- ▶ Improve the planner by tackling its drawbacks
- ▶ Get a domain encoding that suits the planner
- ▶ Acquire knowledge and exploit during planning

## What means?

- ▶ Macro-actions
- ▶ Control-rules
- ▶ Control-policies.

## What Macro-Actions?

- ▶ How to acquire?
- ▶ How to exploit?
- ▶ How to deliver?

How to learn macro-actions in a given context?  
How to make macro-actions readily available in planning?

# How to Achieve Faster Planning?

## When a planner and a domain are given

- ▶ Improve the planner by tackling its drawbacks
- ▶ Get a domain encoding that suits the planner
- ▶ Acquire knowledge and exploit during planning

## What means?

- ▶ Macro-actions
- ▶ Control-rules
- ▶ Control-policies.

## What Macro-Actions?

- ▶ How to acquire?
- ▶ How to exploit?
- ▶ How to deliver?

**How to learn macro-actions in a given context?**  
How to make macro-actions readily available in planning?

# Learning Macro-Actions for Arbitrary Planners and Domains

Hakim Newton, John Levine, Maria Fox, Derek Long

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

International Conference on Automated Planning and Scheduling, 2007

# Outline

## Macro-Actions

Definitions

Implications

## Our Learning Method

Design

Implementation

Experiments

## Conclusions

Related Work

Contribution

Further Work

# 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

## Planning Perspective

- ▶ 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

# 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

## Planning Perspective

- ▶ 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

# Macros: Significance in Planning

## Subplans that are

- ▶ Frequently used
- ▶ Difficult to find
- ▶ Trouble avoiding
- ▶ Trouble recovering

## On the search space

- ▶ Extended visibility of search space.
- + Fewer intermediate states visited.
- ▶ Additional search neighbourhood.
- More branches at search nodes.

## Blocks Domain Example

# Macros: Significance in Planning

## Subplans that are

- ▶ Frequently used
- ▶ Difficult to find
- ▶ Trouble avoiding
- ▶ Trouble recovering

## On the search space

- ▶ Extended visibility of search space.
- + Fewer intermediate states visited.
- ▶ Additional search neighbourhood.
- More branches at search nodes.

## Blocks Domain Example



# Macros: Significance in Planning

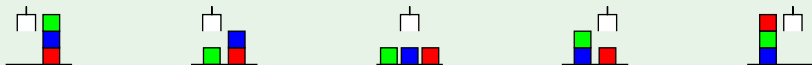
## Subplans that are

- ▶ Frequently used
- ▶ Difficult to find
- ▶ Trouble avoiding
- ▶ Trouble recovering

## On the search space

- ▶ Extended visibility of search space.
- + Fewer intermediate states visited.
- ▶ Additional search neighbourhood.
- More branches at search nodes.

## Blocks Domain Example



# Macros: Significance in Planning

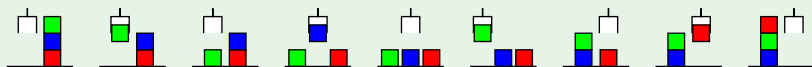
## Subplans that are

- ▶ Frequently used
- ▶ Difficult to find
- ▶ Trouble avoiding
- ▶ Trouble recovering

## On the search space

- ▶ Extended visibility of search space.
- + Fewer intermediate states visited.
- ▶ Additional search neighbourhood.
- More branches at search nodes.

## Blocks Domain Example



# Macros: Significance in Planning

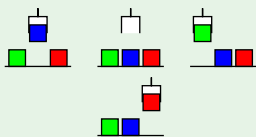
## Subplans that are

- ▶ Frequently used
- ▶ Difficult to find
- ▶ Trouble avoiding
- ▶ Trouble recovering

## On the search space

- ▶ Extended visibility of search space.
- + Fewer intermediate states visited.
- ▶ Additional search neighbourhood.
- More branches at search nodes.

## Blocks Domain Example



# Macros: Significance in Planning

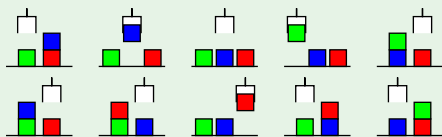
## Subplans that are

- ▶ Frequently used
- ▶ Difficult to find
- ▶ Trouble avoiding
- ▶ Trouble recovering

## On the search space

- ▶ Extended visibility of search space.
- + Fewer intermediate states visited.
- ▶ Additional search neighbourhood.
- More branches at search nodes.

## Blocks Domain Example



# Scope of This Work

Acquisition

Representation

Exploitation

Macro representation and exploitation: **use existing support**

- ▶ 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 formally be recognised in planning.

Macro acquisition: **main objective**

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

# Scope of This Work

Acquisition

Representation

Exploitation

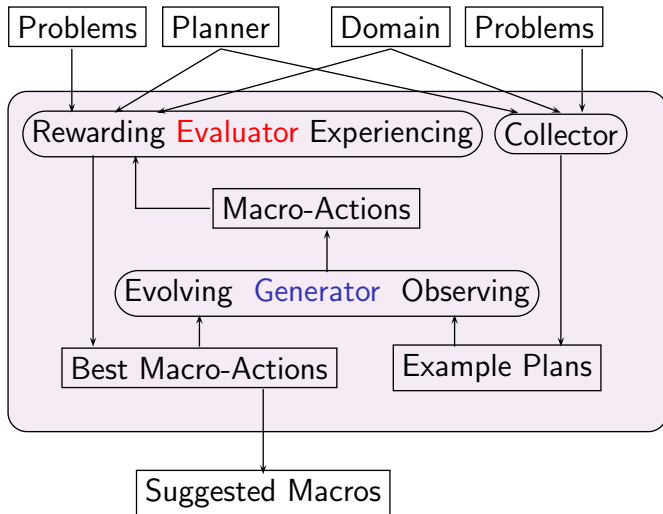
Macro representation and exploitation: **use existing support**

- ▶ 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 formally be recognised in planning.

Macro acquisition: **main objective**

- ▶ Learning individual 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



# Search on Macro Space

## Similar to a genetic algorithm or a multi-point search

1. Generate an initial population of macros and evaluate individuals.
2. For a number of epochs but while offspring generation is possible
  - 2.1 Generate an offspring population and evaluate individuals.
  - 2.2 Replace inferior parents by superior offspring if any.
  - 2.3 Stop if replacement level is unsatisfactory.
3. Suggest best individual macros.

## Dissimilarities to a genetic algorithm

- ▶ Individuals are neither bit-strings nor of equal length.
- ▶ Individuals do not encode system characteristics explicitly.

# Search on Macro Space

## Similar to a genetic algorithm or a multi-point search

1. Generate an initial population of macros and evaluate individuals.
2. For a **number of epochs** but while **offspring generation is possible**
  - 2.1 Generate an offspring population and evaluate individuals.
  - 2.2 Replace inferior parents by superior offspring if any.
  - 2.3 Stop if **replacement level** is unsatisfactory.
3. Suggest best individual macros.

## Dissimilarities to a genetic algorithm

- ▶ Individuals are neither bit-strings nor of equal length.
- ▶ Individuals do not encode system characteristics explicitly.

# Macro Representation

## Action sequence form: used for manipulation

```
(:macro  
  (move ?ra ?rb)  
  (pick ?b ?rb ?g)  
  (move ?rb ?ra)  
)
```

## Compiled form: used by current planners

```
(:action move-pick-move  
  :parameters (?ra ?rb - room ?b - ball ?g - gripper)  
  :precondition (and  
    (at-robby ?ra)(at ?b ?rb)(free ?g)  
  )  
  :effect (and  
    (carry ?b ?g)(not (at ?b ?rb))(not (free ?g))  
  )  
)
```

# Macro Representation

Action sequence form: used for manipulation

```
(:macro  
  (move ?ra ?rb)  
  (pick ?b ?rb ?g)  
  (move ?rb ?ra)  
)
```

Compiled form: used by current planners

```
(:action move-pick-move  
  :parameters (?ra ?rb - room ?b - ball ?g - gripper)  
  :precondition (and  
    (at-robby ?ra)(at ?b ?rb)(free ?g)  
  )  
  :effect (and  
    (carry ?b ?g)(not (at ?b ?rb))(not (free ?g))  
  )  
)
```

# Example Problems

## Selection Criteria

*Knowledge should be acquired from simpler situations, reinforced in complex but manageable situations, and applied in yet more complex and even unmanageable situations.*

## Problem Size: in solution time within resource limits

- ▶ For generation, smaller and solvable; to build up a plan library
- ▶ For evaluation, small and solvable; solved for every macro
- ▶ For demonstration, large and may be unsolvable
- ▶ Larger problems get more weight whenever possible

# Example Problems

## Selection Criteria

*Knowledge should be acquired from simpler situations, reinforced in complex but manageable situations, and applied in yet more complex and even unmanageable situations.*

## Problem Size: in solution time within resource limits

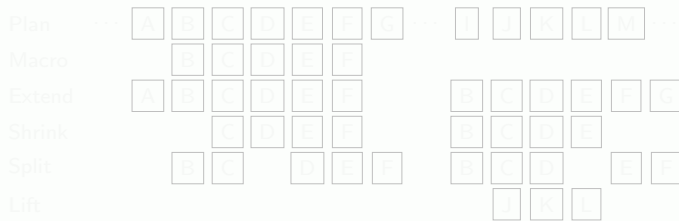
- ▶ For generation, smaller and solvable; to build up a plan library
- ▶ For evaluation, small and solvable; solved for every macro
- ▶ For demonstration, large and may be unsolvable
- ▶ Larger problems get more weight whenever possible

# Macro Generation

## Macros occurring in plans only

- ▶ Plans are footprints of the planner on the domain landscape.
- ▶ Plans inherently bear properties esp. that lead to the solution.

## Operators



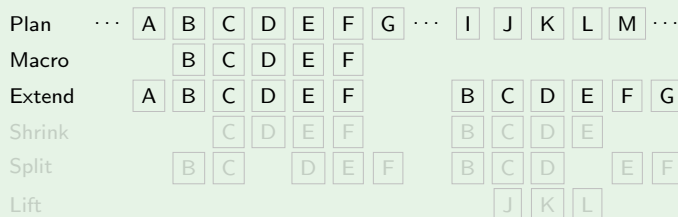


# Macro Generation

## Macros occurring in plans only

- ▶ Plans are footprints of the planner on the domain landscape.
- ▶ Plans inherently bear properties esp. that lead to the solution.

## Operators





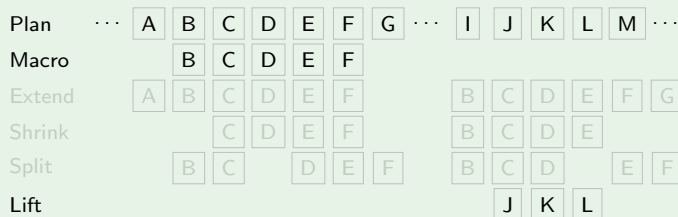


# Macro Generation

## Macros occurring in plans only

- ▶ Plans are footprints of the planner on the domain landscape.
- ▶ Plans inherently bear properties esp. that lead to the solution.

## Operators



# Macro Evaluation

## Evaluation Criteria

*For a good macro, **most problems (Cover)** should be solved taking **less time (Score)** in **most cases (Point)** with its augmented domain (compared to the original domain).*

## Utility Formula

- ▶ Cover = % problems solved
- ▶ Score = Weighted mean time gain
- ▶ Point = % problems solved faster
- ▶ **Utility = Cover × Score × Point**

# Macro Pruning and Validation

## Pruning

- ▶ Null and inconsistent effect; unsatisfiable precondition
- ▶ Limits on parameter count and macro length
- ▶ Duplication and equivalence of macros
- ▶ Cohesion type: common variables, not causal links
- ▶ Runtime failure to plan within limits

## Validation

- ▶ Planners sometimes produce invalid plans with macros.
- ▶ Plans with macros are therefore validated as needed.

## Results: Summary Table

<i>domain-planner-macro</i>	+S -s	+T -t	P ± p	+L -l	Q ± q
Blocks-FF-1	+36 -0	+60 -4	65 ± 19	+58 -4	20 ± 2
Blocks-LPG-1	+0 -0	+94 -0	58 ± 2	+92 -0	28 ± 2
Blocks-SGPlan-1	+6 -0	+72 -12	-19 ± 76	+38 -34	-2 ± 2
Blocks-VHPOP-1	+54 -0	+26 -2	79 ± 9	+0 -2	-2 ± 2
Ferry-FD-1	+0 -0	+100 -0	93 ± 0	+96 -4	9 ± 0
Ferry-LPG-1	+0 -0	+92 -0	92 ± 0	+0 -92	-18 ± 0
Ferry-SatPlan-1	+10 -0	+88 -2	94 ± 2	+0 -90	-64 ± 5
Ferry-VHPOP-1	+68 -0	+32 -0	100 ± 0	+0 -32	-26 ± 2
Gripper-FD-1	+54 -0	+46 -0	88 ± 1	+0 -46	0 ± 0
Gripper-LPG-1	+0 -0	+100 -0	92 ± 0	+0 -100	-23 ± 0
Gripper-SatPlan-1	+20 -0	+78 -2	77 ± 4	+0 -80	-47 ± 1
Gripper-VHPOP-1	+36 -0	+64 -0	85 ± 4	+0 -36	-17 ± 3
Satellite-FF-1	+0 -0	+100 -0	96 ± 0	+0 -100	-43 ± 1
Satellite-VHPOP-1	+12 -0	+22 -6	51 ± 21	+0 -28	-13 ± 2
NFerry-FF-1	+32 -0	+62 -6	66 ± 6	+0 -68	-85 ± 3
NFerry-SGPlan-1	+4 -0	+46 -36	33 ± 8	+14 -68	-8 ± 1
NSatellite-FF-1	+58 -0	+42 -0	98 ± 0	+0 -42	-10 ± 1
NSatellite-LPG-1	+64 -0	+38 -0	48 ± 3	+0 -18	-11 ± 1
NSatellite-SGPlan-1	+0 -0	+100 -0	39 ± 1	+0 -100	-29 ± 1

## Results: Summary Table Solvability: Gain(+S), Loss(-s)

<i>domain-planner-macro</i>	+S -s	+T -t	P ± p	+L -l	Q ± q
Blocks-FF-1	+36 -0	+60 -4	65 ± 19	+58 -4	20 ± 2
Blocks-LPG-1	+0 -0	+94 -0	58 ± 2	+92 -0	28 ± 2
Blocks-SGPlan-1	+6 -0	+72 -12	-19 ± 76	+38 -34	-2 ± 2
Blocks-VHPOP-1	+54 -0	+26 -2	79 ± 9	+0 -2	-2 ± 2
Ferry-FD-1	+0 -0	+100 -0	93 ± 0	+96 -4	9 ± 0
Ferry-LPG-1	+0 -0	+92 -0	92 ± 0	+0 -92	-18 ± 0
Ferry-SatPlan-1	+10 -0	+88 -2	94 ± 2	+0 -90	-64 ± 5
Ferry-VHPOP-1	+68 -0	+32 -0	100 ± 0	+0 -32	-26 ± 2
Gripper-FD-1	+54 -0	+46 -0	88 ± 1	+0 -46	0 ± 0
Gripper-LPG-1	+0 -0	+100 -0	92 ± 0	+0 -100	-23 ± 0
Gripper-SatPlan-1	+20 -0	+78 -2	77 ± 4	+0 -80	-47 ± 1
Gripper-VHPOP-1	+36 -0	+64 -0	85 ± 4	+0 -36	-17 ± 3
Satellite-FF-1	+0 -0	+100 -0	96 ± 0	+0 -100	-43 ± 1
Satellite-VHPOP-1	+12 -0	+22 -6	51 ± 21	+0 -28	-13 ± 2
NFerry-FF-1	+32 -0	+62 -6	66 ± 6	+0 -68	-85 ± 3
NFerry-SGPlan-1	+4 -0	+46 -36	33 ± 8	+14 -68	-8 ± 1
NSatellite-FF-1	+58 -0	+42 -0	98 ± 0	+0 -42	-10 ± 1
NSatellite-LPG-1	+64 -0	+38 -0	48 ± 3	+0 -18	-11 ± 1
NSatellite-SGPlan-1	+0 -0	+100 -0	39 ± 1	+0 -100	-29 ± 1

## Results: Summary Table Time Gain +T, Loss -t probs

<i>domain-planner-macro</i>	+S -s	+T -t	P ± p	+L -l	Q ± q
Blocks-FF-1	+36 -0	+60 -4	65 ± 19	+58 -4	20 ± 2
Blocks-LPG-1	+0 -0	+94 -0	58 ± 2	+92 -0	28 ± 2
Blocks-SGPlan-1	+6 -0	+72 -12	-19 ± 76	+38 -34	-2 ± 2
Blocks-VHPOP-1	+54 -0	+26 -2	79 ± 9	+0 -2	-2 ± 2
Ferry-FD-1	+0 -0	+100 -0	93 ± 0	+96 -4	9 ± 0
Ferry-LPG-1	+0 -0	+92 -0	92 ± 0	+0 -92	-18 ± 0
Ferry-SatPlan-1	+10 -0	+88 -2	94 ± 2	+0 -90	-64 ± 5
Ferry-VHPOP-1	+68 -0	+32 -0	100 ± 0	+0 -32	-26 ± 2
Gripper-FD-1	+54 -0	+46 -0	88 ± 1	+0 -46	0 ± 0
Gripper-LPG-1	+0 -0	+100 -0	92 ± 0	+0 -100	-23 ± 0
Gripper-SatPlan-1	+20 -0	+78 -2	77 ± 4	+0 -80	-47 ± 1
Gripper-VHPOP-1	+36 -0	+64 -0	85 ± 4	+0 -36	-17 ± 3
Satellite-FF-1	+0 -0	+100 -0	96 ± 0	+0 -100	-43 ± 1
Satellite-VHPOP-1	+12 -0	+22 -6	51 ± 21	+0 -28	-13 ± 2
NFerry-FF-1	+32 -0	+62 -6	66 ± 6	+0 -68	-85 ± 3
NFerry-SGPlan-1	+4 -0	+46 -36	33 ± 8	+14 -68	-8 ± 1
NSatellite-FF-1	+58 -0	+42 -0	98 ± 0	+0 -42	-10 ± 1
NSatellite-LPG-1	+64 -0	+38 -0	48 ± 3	+0 -18	-11 ± 1
NSatellite-SGPlan-1	+0 -0	+100 -0	39 ± 1	+0 -100	-29 ± 1

Results: Summary Table Time Gain Mean(P) $\pm$ Disp(p)

<i>domain-planner-macro</i>	+S -s	+T -t	P $\pm$ p	+L -l	Q $\pm$ q
Blocks-FF-1	+36 -0	+60 -4	65 $\pm$ 19	+58 -4	20 $\pm$ 2
Blocks-LPG-1	+0 -0	+94 -0	58 $\pm$ 2	+92 -0	28 $\pm$ 2
Blocks-SGPlan-1	+6 -0	+72 -12	-19 $\pm$ 76	+38 -34	-2 $\pm$ 2
Blocks-VHPOP-1	+54 -0	+26 -2	79 $\pm$ 9	+0 -2	-2 $\pm$ 2
Ferry-FD-1	+0 -0	+100 -0	93 $\pm$ 0	+96 -4	9 $\pm$ 0
Ferry-LPG-1	+0 -0	+92 -0	92 $\pm$ 0	+0 -92	-18 $\pm$ 0
Ferry-SatPlan-1	+10 -0	+88 -2	94 $\pm$ 2	+0 -90	-64 $\pm$ 5
Ferry-VHPOP-1	+68 -0	+32 -0	100 $\pm$ 0	+0 -32	-26 $\pm$ 2
Gripper-FD-1	+54 -0	+46 -0	88 $\pm$ 1	+0 -46	0 $\pm$ 0
Gripper-LPG-1	+0 -0	+100 -0	92 $\pm$ 0	+0 -100	-23 $\pm$ 0
Gripper-SatPlan-1	+20 -0	+78 -2	77 $\pm$ 4	+0 -80	-47 $\pm$ 1
Gripper-VHPOP-1	+36 -0	+64 -0	85 $\pm$ 4	+0 -36	-17 $\pm$ 3
Satellite-FF-1	+0 -0	+100 -0	96 $\pm$ 0	+0 -100	-43 $\pm$ 1
Satellite-VHPOP-1	+12 -0	+22 -6	51 $\pm$ 21	+0 -28	-13 $\pm$ 2
NFerry-FF-1	+32 -0	+62 -6	66 $\pm$ 6	+0 -68	-85 $\pm$ 3
NFerry-SGPlan-1	+4 -0	+46 -36	33 $\pm$ 8	+14 -68	-8 $\pm$ 1
NSatellite-FF-1	+58 -0	+42 -0	98 $\pm$ 0	+0 -42	-10 $\pm$ 1
NSatellite-LPG-1	+64 -0	+38 -0	48 $\pm$ 3	+0 -18	-11 $\pm$ 1
NSatellite-SGPlan-1	+0 -0	+100 -0	39 $\pm$ 1	+0 -100	-29 $\pm$ 1

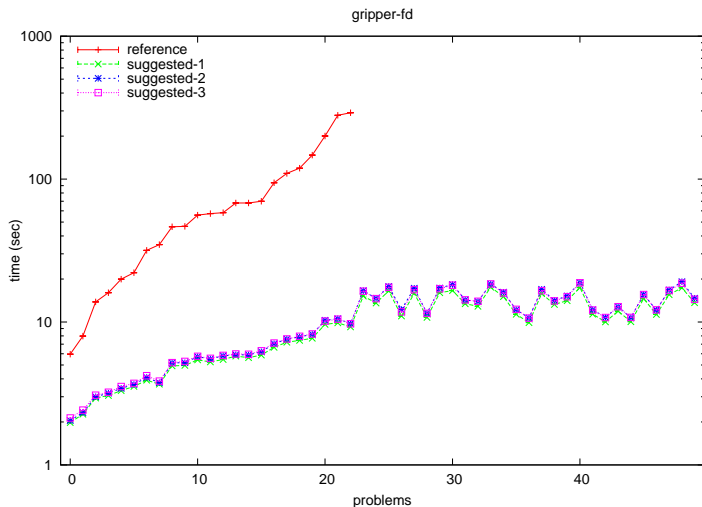
## Results: Summary Table Length Gain +L, Loss -l probs

<i>domain-planner-macro</i>	+S -s	+T -t	P ± p	+L -l	Q ± q
Blocks-FF-1	+36 -0	+60 -4	65 ± 19	+58 -4	20 ± 2
Blocks-LPG-1	+0 -0	+94 -0	58 ± 2	+92 -0	28 ± 2
Blocks-SGPlan-1	+6 -0	+72 -12	-19 ± 76	+38 -34	-2 ± 2
Blocks-VHPOP-1	+54 -0	+26 -2	79 ± 9	+0 -2	-2 ± 2
Ferry-FD-1	+0 -0	+100 -0	93 ± 0	+96 -4	9 ± 0
Ferry-LPG-1	+0 -0	+92 -0	92 ± 0	+0 -92	-18 ± 0
Ferry-SatPlan-1	+10 -0	+88 -2	94 ± 2	+0 -90	-64 ± 5
Ferry-VHPOP-1	+68 -0	+32 -0	100 ± 0	+0 -32	-26 ± 2
Gripper-FD-1	+54 -0	+46 -0	88 ± 1	+0 -46	0 ± 0
Gripper-LPG-1	+0 -0	+100 -0	92 ± 0	+0 -100	-23 ± 0
Gripper-SatPlan-1	+20 -0	+78 -2	77 ± 4	+0 -80	-47 ± 1
Gripper-VHPOP-1	+36 -0	+64 -0	85 ± 4	+0 -36	-17 ± 3
Satellite-FF-1	+0 -0	+100 -0	96 ± 0	+0 -100	-43 ± 1
Satellite-VHPOP-1	+12 -0	+22 -6	51 ± 21	+0 -28	-13 ± 2
NFerry-FF-1	+32 -0	+62 -6	66 ± 6	+0 -68	-85 ± 3
NFerry-SGPlan-1	+4 -0	+46 -36	33 ± 8	+14 -68	-8 ± 1
NSatellite-FF-1	+58 -0	+42 -0	98 ± 0	+0 -42	-10 ± 1
NSatellite-LPG-1	+64 -0	+38 -0	48 ± 3	+0 -18	-11 ± 1
NSatellite-SGPlan-1	+0 -0	+100 -0	39 ± 1	+0 -100	-29 ± 1

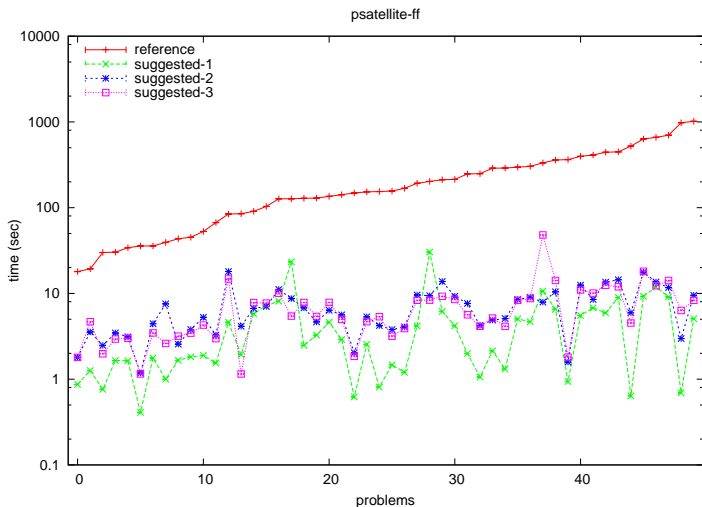
## Results: Summary Table Length Gain Mean(Q)±Disp(q)

<i>domain-planner-macro</i>	+S -s	+T -t	P ± p	+L -l	Q ± q
Blocks-FF-1	+36 -0	+60 -4	65 ± 19	+58 -4	20 ± 2
Blocks-LPG-1	+0 -0	+94 -0	58 ± 2	+92 -0	28 ± 2
Blocks-SGPlan-1	+6 -0	+72 -12	-19 ± 76	+38 -34	-2 ± 2
Blocks-VHPOP-1	+54 -0	+26 -2	79 ± 9	+0 -2	-2 ± 2
Ferry-FD-1	+0 -0	+100 -0	93 ± 0	+96 -4	9 ± 0
Ferry-LPG-1	+0 -0	+92 -0	92 ± 0	+0 -92	-18 ± 0
Ferry-SatPlan-1	+10 -0	+88 -2	94 ± 2	+0 -90	-64 ± 5
Ferry-VHPOP-1	+68 -0	+32 -0	100 ± 0	+0 -32	-26 ± 2
Gripper-FD-1	+54 -0	+46 -0	88 ± 1	+0 -46	0 ± 0
Gripper-LPG-1	+0 -0	+100 -0	92 ± 0	+0 -100	-23 ± 0
Gripper-SatPlan-1	+20 -0	+78 -2	77 ± 4	+0 -80	-47 ± 1
Gripper-VHPOP-1	+36 -0	+64 -0	85 ± 4	+0 -36	-17 ± 3
Satellite-FF-1	+0 -0	+100 -0	96 ± 0	+0 -100	-43 ± 1
Satellite-VHPOP-1	+12 -0	+22 -6	51 ± 21	+0 -28	-13 ± 2
NFerry-FF-1	+32 -0	+62 -6	66 ± 6	+0 -68	-85 ± 3
NFerry-SGPlan-1	+4 -0	+46 -36	33 ± 8	+14 -68	-8 ± 1
NSatellite-FF-1	+58 -0	+42 -0	98 ± 0	+0 -42	-10 ± 1
NSatellite-LPG-1	+64 -0	+38 -0	48 ± 3	+0 -18	-11 ± 1
NSatellite-SGPlan-1	+0 -0	+100 -0	39 ± 1	+0 -100	-29 ± 1

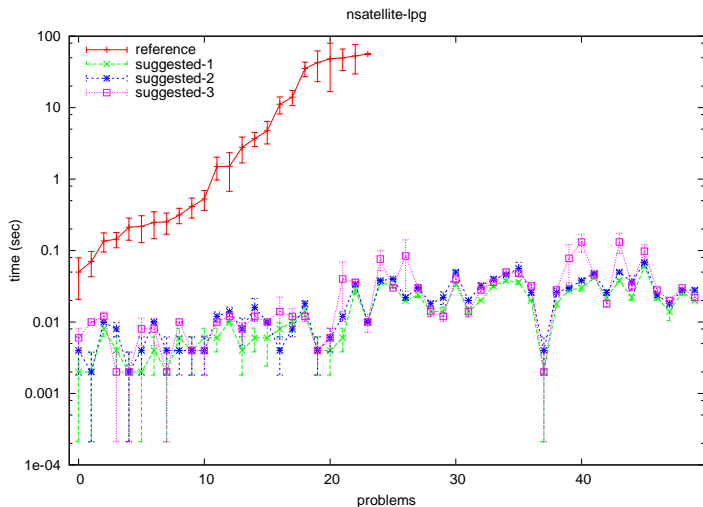
# Results: Gripper Fast-Downward



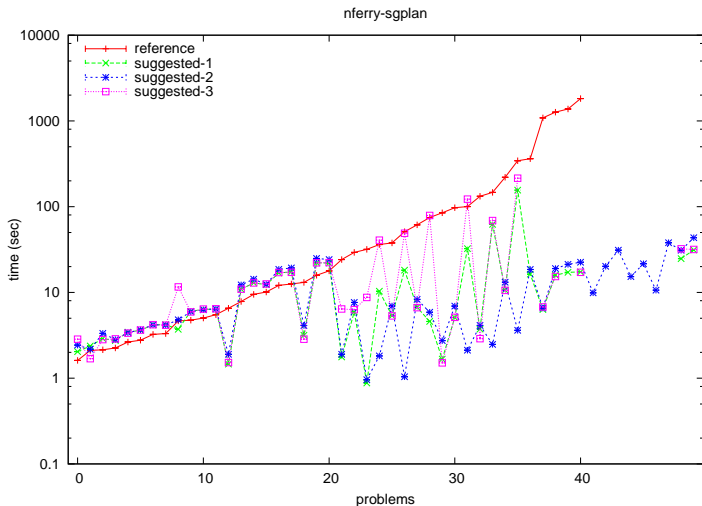
# Results: PSatellite Fast-Forward



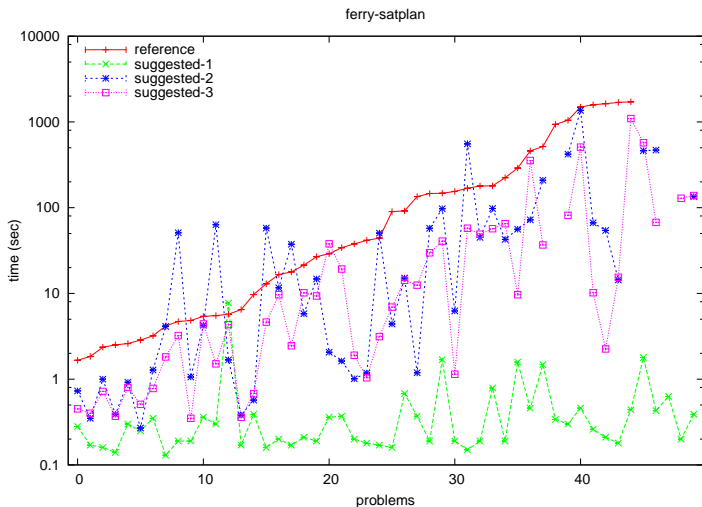
# Results: NSatellite LPG



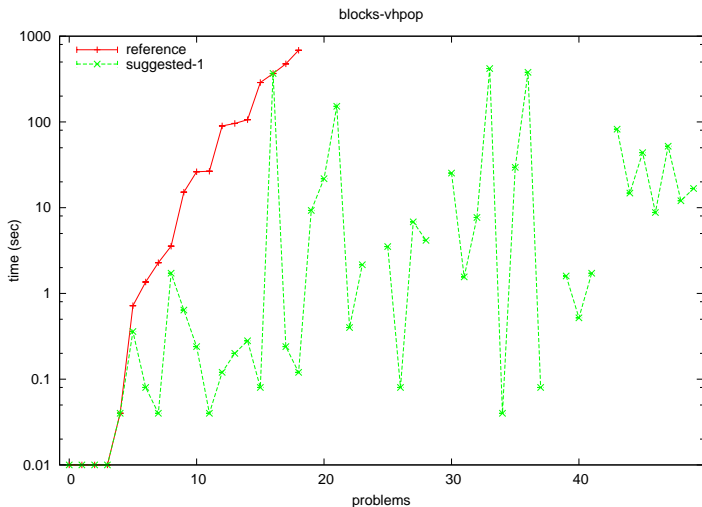
# Results: NFerry SGPlan



# Results: Ferry SatPlan



# Results: Blocks VHPOP



# Analysis

## Hypotheses

1. Consistency of the evaluation method
2. Effectiveness of macro learning from plans
3. Success of learning macros using no property
4. Success of capturing various types of macros

## Training Time: parameters to be tuned

hours	fd	ff	lpg	satplan	sgplan	vhpop
blocks	14.5	19.5	58.7	3.1	9.1	0.5
ferry	17.2	15.4	32.2	1.0	19.1	0.3
gripper	4.1	3.1	21.2	1.1	3.0	0.7
satellite	31.2	24.3	152.4	0.8	13.7	1.2
nferry		23.0			35.6	
nsatellite		38.4	30.0		27.0	

CPU 3GHZ

RAM 2GB

RankTime 10s

RankMem 1GB

EpochLim 100

PopSize 2actns

Replc 1/25epch

# Analysis

## Hypotheses

1. Consistency of the evaluation method
2. Effectiveness of macro learning from plans
3. Success of learning macros using no property
4. Success of capturing various types of macros

## Training Time: parameters to be tuned

hours	fd	ff	lpg	satplan	sgplan	vhpop
blocks	14.5	19.5	58.7	3.1	9.1	0.5
ferry	17.2	15.4	32.2	1.0	19.1	0.3
gripper	4.1	3.1	21.2	1.1	3.0	0.7
satellite	31.2	24.3	152.4	0.8	13.7	1.2
nferry		23.0			35.6	
nsatellite		38.4	30.0		27.0	

CPU 3GHZ  
RAM 2GB  
RankTime 10s  
RankMem 1GB  
EpochLim 100  
PopSize 2actns  
Replc 1/25epch

## 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
Don't care; don't know	Any planner	Wizard
Symmetry, almost symmetry	Briefcase, Gripper, etc.	MARVIN
Component abstraction	Gripper, Rover, etc.	Macro-FF
Operator decomposability	Rubik's cube, etc.	Korf's MPS
Don't care; don't know	Any domain	Wizard

- ▶ Unlike existing methods, Wizard deals with acquisition part only.
- ▶ Unlike Wizard, they have no search guidance on macro space.

# Contribution

## Existing Methods

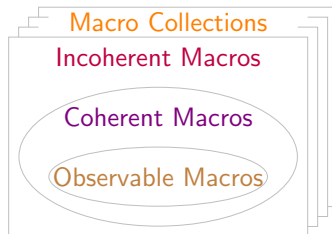
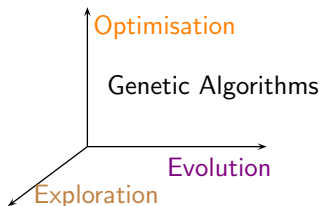
- ▶ Acquire macros from specific domain or planner properties
- ▶ Are actually planners with limited macro learning capability

## Our Method

- ▶ Macro acquisition to improve performance
- ▶ No modification to representation or exploitation component
- ▶ Works readily for arbitrary planners and domains
- ▶ Does not use any planner or domain properties
- ▶ Is not a planner at all, rather a learner that suggests macros



# Our Further Work



## Exhaustive use of genetic algorithms on macros

- ▶ This work explores observable macros from given plans.
- ▶ We would evolve non-observable coherent macros.
- ▶ We would evolve incoherent and catalytic macros.
- ▶ We would optimise collections of macros to be suggested.

# 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.

## How macros evolve in Blocks for FF

*Was observable*

(unstack ?b5 ?b0)  
(stack ?b5 ?b2)  
(pick ?b0)

*Was non-observable*

(unstack ?b5 ?b0)  
(stack ?b5 ?b2)  
(pick ?b0)  
(stack ?b0 ?b5)

*Incoherent*

(drop ?b5)  
(unstack ?b5 ?b0)  
(stack ?b5 ?b2)  
(pick ?b0)  
(stack ?b0 ?b5)

# 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.

## How macros evolve in Blocks for FF

*Was observable*

```
(unstack ?b5 ?b0)  
(stack ?b5 ?b2)  
(pick ?b0)
```

*Was non-observable*

```
(unstack ?b5 ?b0)  
(stack ?b5 ?b2)  
(pick ?b0)  
(stack ?b0 ?b5)
```

*Incoherent*

```
(drop ?b5)  
(unstack ?b5 ?b0)  
(stack ?b5 ?b2)  
(pick ?b0)  
(stack ?b0 ?b5)
```

