

# Combinations of Domain Enhancing Macro-Actions in Planning

Hakim Newton, John Levine

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

General Workshop on AI, EPIA'2007

# Outline

## Introduction

Planning

Macro-Actions

## Our Learning Method

Design

Implementation

Experiments

## Conclusions

Related Work

Contribution

Further Work

# Planning

## Definitions and Terminologies

- ▶ Given a world (or **domain**), find a sequence of actions (or **plan**) that achieves a **goal state** starting from an **initial state**.
- ▶ A planning method (or **planner**) should work with any domain.

## Blocks World Example

- ▶ Blocks arranged in stacks on a table are to be re-stacked.
- ▶ Allowed actions are Pick, Drop, Stack, Unstack.

# Planning

## Definitions and Terminologies

- ▶ Given a world (or **domain**), find a sequence of actions (or **plan**) that achieves a **goal state** starting from an **initial state**.
- ▶ A planning method (or **planner**) should work with any domain.

## Blocks World Example

- ▶ Blocks arranged in stacks on a table are to be re-stacked.
- ▶ Allowed actions are Pick, Drop, Stack, Unstack.



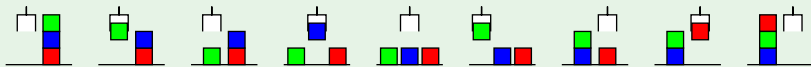
# Planning

## Definitions and Terminologies

- ▶ Given a world (or **domain**), find a sequence of actions (or **plan**) that achieves a **goal state** starting from an **initial state**.
- ▶ A planning method (or **planner**) should work with any domain.

## Blocks World Example

- ▶ Blocks arranged in stacks on a table are to be re-stacked.
- ▶ Allowed actions are Pick, Drop, Stack, Unstack.



# Planning Domain Definition Language

```
(domain blocksworld)
  (:action pick :parameters (?b)
   :precondition (and
    (top-clear ?b)
    (on-table ?b)
    (arm-free)
   )
   :effect (and
    (not (top-clear ?b))
    (not (on-table ?b))
    (not (arm-free))
    (arm-holding ?b)
   )
  )
  (:action drop :parameters (?b)
  )
  (:action unstack :parameters (?ab ?bb)
  )
  (:action stack :parameters (?ab ?bb)
  )
)
```

```
(problem blocksworld-3)
  (:init
   (arm-free)
   (on-table red)
   (on blue red)
   (on green blue)
   (top-clear green)
  )
  (:goal
   (arm-free)
   (on-table blue)
   (on green blue)
   (on red green)
   (top-clear red)
  )
)
(plan blocksworld-3)
  (unstack green) (drop green)
  (unstack blue) (drop blue)
  (pick green) (stack green blue)
  (pick red) (stack red green)
)
```

## 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 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 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 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 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?

# 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 concept in planning language PDDL
- ▶ Planners are not aware of or reason about macro-actions
- ▶ Only standard way to reason: macros in disguise of actions

# 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 concept in planning language PDDL
- ▶ Planners are not aware of or reason about macro-actions
- ▶ Only standard way to reason: macros in disguise of actions

## 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

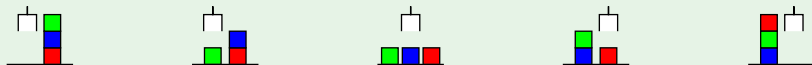
## Subplans that are

- ▶ 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

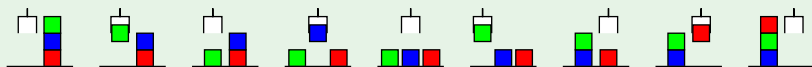
## Subplans that are

- ▶ 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

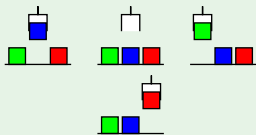
## Subplans that are

- ▶ 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

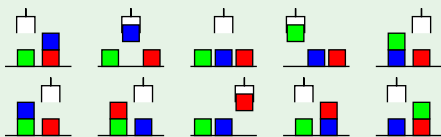
## Subplans that are

- ▶ 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**

- ▶ 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 macro collections for arbitrary planners and domains
- ▶ One macro may not suffice; Macros may interact among them
- ▶ Arbitrariness is in structural properties, not just PDDL levels
- ▶ Improving performance making no modification to a planner

# Scope of This Work

Acquisition

Representation

Exploitation

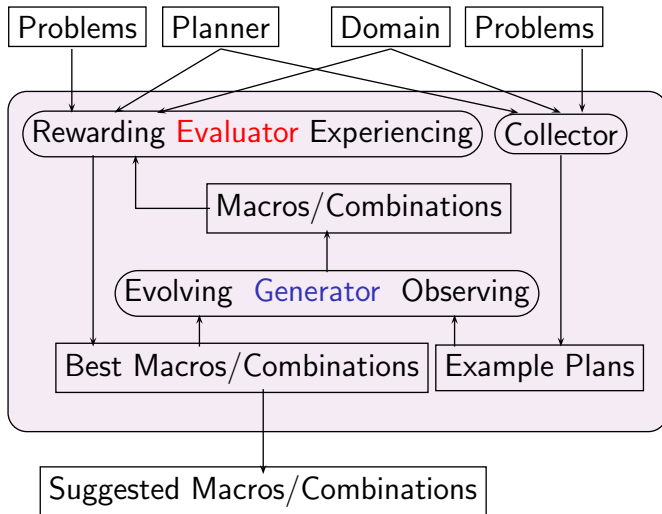
Macro representation and exploitation: use existing support

- ▶ 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 macro collections for arbitrary planners and domains
- ▶ One macro may not suffice; Macros may interact among them
- ▶ Arbitrariness is in structural properties, not just PDDL levels
- ▶ Improving performance making no modification to a planner

# System Architecture



# Search on Macro Space then on Combination Space

Similar to a genetic algorithm or a multi-point search

1. Generate an initial population 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 individuals.

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 then on Combination Space

Similar to a genetic algorithm or a multi-point search

1. Generate an initial population 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 individuals.

Dissimilarities to a genetic algorithm

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

# Macro/Combination Representation

## Composite forms: used for manipulation

<pre>(:macro   (move ?ra ?rb)   (pick ?b ?rb ?g)   (move ?rb ?ra) )</pre>	<pre>(:combination   macro-1   macro-2   macro-3 )</pre>
---	--

## 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/Combination Representation

## Composite forms: used for manipulation

<pre>(:macro</pre>	<pre>(:combination</pre>
<pre>  (move ?ra ?rb)</pre>	<pre>  macro-1</pre>
<pre>  (pick ?b ?rb ?g)</pre>	<pre>  macro-2</pre>
<pre>  (move ?rb ?ra)</pre>	<pre>  macro-3</pre>
<pre>)</pre>	<pre>)</pre>

## 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

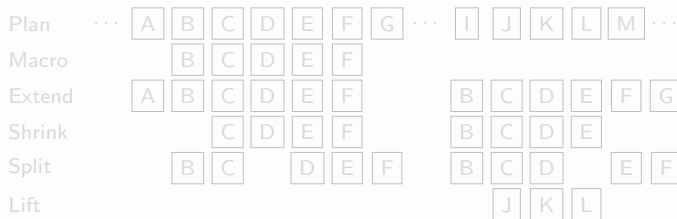
- ▶ For generation, smaller and solvable; to build up a plan library
- ▶ For evaluation, small and solvable; solved for every macro
- ▶ For demonstration, large and may be unsolvable
- ▶ Larger problems get more weight whenever possible

# 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

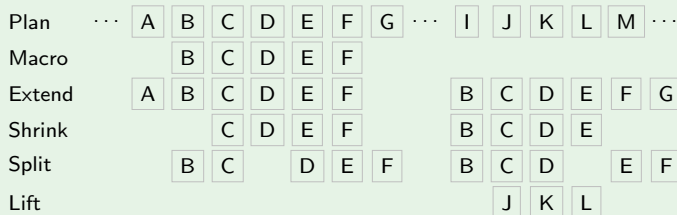


# Macro Generation

## 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 Combination Generation

## Operands

- ▶ Individual macros are explored in the first run
- ▶ Combinations of macros are explored in the second run

## Operators

Bunches

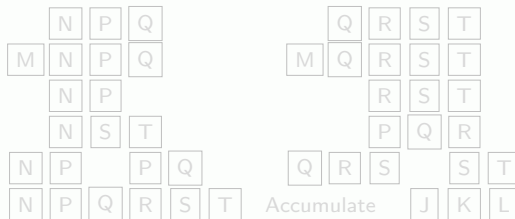
Add M

Delete Q

Crossover Q

Split

Merge

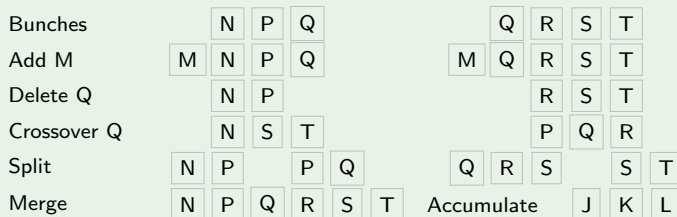


# Macro Combination Generation

## Operands

- ▶ Individual macros are explored in the first run
- ▶ Combinations of macros are explored in the second run

## Operators



# Macro/Combination Evaluation

## Evaluation Criteria

*For a good macro/combination, 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 Macro Count in Collections

<i>domain-planner</i>	N	+S -s	+T -t	P ± p	+L -l	Q ± q
Blocks-FF	3	+40 -0	+56 -0	92 ± 3	+52 -6	22 ± 2
Blocks-LPG	2	+0 -0	+98 -0	63 ± 1	+98 -0	49 ± 1
Blocks-SGPLAN	3	+12 -0	+70 -14	34 ± 14	+60 -26	9 ± 3
Gripper-FF	2	+0 -0	+100 -0	97 ± 0	+0 -100	-31 ± 0
Gripper-LPG	2	+0 -0	+100 -0	90 ± 0	+0 -100	-22 ± 0
Gripper-VHPOP	2	+34 -0	+66 -0	99 ± 0	+0 -56	-19 ± 1
Ferry-FF	2	+0 -6	+94 -0	76 ± 1	+0 -94	-70 ± 3
Ferry-LPG	2	+0 -0	+86 -0	89 ± 0	+0 -86	-21 ± 0
Ferry-SATPLAN	2	+16 -0	+84 -0	93 ± 2	+0 -84	-63 ± 4
Ferry-VHPOP	3	+76 -0	+22 -0	100 ± 0	+0 -22	-25 ± 3
NFerry-FF	2	+16 -4	+70 -0	54 ± 1	+0 -70	-5 ± 0
NFerry-SGPLAN	3	+16 -0	+52 -32	24 ± 13	+2 -82	-18 ± 1
Satellite-FF	2	+2 -34	+34 -0	95 ± 0	+0 -34	-45 ± 1
Satellite-VHPOP	4	+50 -0	+28 -0	91 ± 4	+0 -20	-14 ± 3
NSatellite-FF	2	+4 -8	+30 -0	99 ± 0	+4 -26	-8 ± 1
NSatellite-LPG	3	+62 -0	+42 -0	50 ± 4	+0 -36	-13 ± 1
NSatellite-SGPLAN	2	+0 -0	+100 -0	37 ± 0	+0 -100	-28 ± 1

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

<i>domain-planner</i>	N	+S -s	+T -t	P ± p	+L -l	Q ± q
Blocks-FF	3	+40 -0	+56 -0	92 ± 3	+52 -6	22 ± 2
Blocks-LPG	2	+0 -0	+98 -0	63 ± 1	+98 -0	49 ± 1
Blocks-SGPLAN	3	+12 -0	+70 -14	34 ± 14	+60 -26	9 ± 3
Gripper-FF	2	+0 -0	+100 -0	97 ± 0	+0 -100	-31 ± 0
Gripper-LPG	2	+0 -0	+100 -0	90 ± 0	+0 -100	-22 ± 0
Gripper-VHPOP	2	+34 -0	+66 -0	99 ± 0	+0 -56	-19 ± 1
Ferry-FF	2	+0 -6	+94 -0	76 ± 1	+0 -94	-70 ± 3
Ferry-LPG	2	+0 -0	+86 -0	89 ± 0	+0 -86	-21 ± 0
Ferry-SATPLAN	2	+16 -0	+84 -0	93 ± 2	+0 -84	-63 ± 4
Ferry-VHPOP	3	+76 -0	+22 -0	100 ± 0	+0 -22	-25 ± 3
NFerry-FF	2	+16 -4	+70 -0	54 ± 1	+0 -70	-5 ± 0
NFerry-SGPLAN	3	+16 -0	+52 -32	24 ± 13	+2 -82	-18 ± 1
Satellite-FF	2	+2 -34	+34 -0	95 ± 0	+0 -34	-45 ± 1
Satellite-VHPOP	4	+50 -0	+28 -0	91 ± 4	+0 -20	-14 ± 3
NSatellite-FF	2	+4 -8	+30 -0	99 ± 0	+4 -26	-8 ± 1
NSatellite-LPG	3	+62 -0	+42 -0	50 ± 4	+0 -36	-13 ± 1
NSatellite-SGPLAN	2	+0 -0	+100 -0	37 ± 0	+0 -100	-28 ± 1

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

<i>domain-planner</i>	N	+S -s	+T -t	P $\pm$ p	+L -l	Q $\pm$ q
Blocks-FF	3	+40 -0	+56 -0	92 $\pm$ 3	+52 -6	22 $\pm$ 2
Blocks-LPG	2	+0 -0	+98 -0	63 $\pm$ 1	+98 -0	49 $\pm$ 1
Blocks-SGPLAN	3	+12 -0	+70 -14	34 $\pm$ 14	+60 -26	9 $\pm$ 3
Gripper-FF	2	+0 -0	+100 -0	97 $\pm$ 0	+0 -100	-31 $\pm$ 0
Gripper-LPG	2	+0 -0	+100 -0	90 $\pm$ 0	+0 -100	-22 $\pm$ 0
Gripper-VHPOP	2	+34 -0	+66 -0	99 $\pm$ 0	+0 -56	-19 $\pm$ 1
Ferry-FF	2	+0 -6	+94 -0	76 $\pm$ 1	+0 -94	-70 $\pm$ 3
Ferry-LPG	2	+0 -0	+86 -0	89 $\pm$ 0	+0 -86	-21 $\pm$ 0
Ferry-SATPLAN	2	+16 -0	+84 -0	93 $\pm$ 2	+0 -84	-63 $\pm$ 4
Ferry-VHPOP	3	+76 -0	+22 -0	100 $\pm$ 0	+0 -22	-25 $\pm$ 3
NFerry-FF	2	+16 -4	+70 -0	54 $\pm$ 1	+0 -70	-5 $\pm$ 0
NFerry-SGPLAN	3	+16 -0	+52 -32	24 $\pm$ 13	+2 -82	-18 $\pm$ 1
Satellite-FF	2	+2 -34	+34 -0	95 $\pm$ 0	+0 -34	-45 $\pm$ 1
Satellite-VHPOP	4	+50 -0	+28 -0	91 $\pm$ 4	+0 -20	-14 $\pm$ 3
NSatellite-FF	2	+4 -8	+30 -0	99 $\pm$ 0	+4 -26	-8 $\pm$ 1
NSatellite-LPG	3	+62 -0	+42 -0	50 $\pm$ 4	+0 -36	-13 $\pm$ 1
NSatellite-SGPLAN	2	+0 -0	+100 -0	37 $\pm$ 0	+0 -100	-28 $\pm$ 1

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

<i>domain-planner</i>	N	+S -s	+T -t	P $\pm$ p	+L -l	Q $\pm$ q
Blocks-FF	3	+40 -0	+56 -0	92 $\pm$ 3	+52 -6	22 $\pm$ 2
Blocks-LPG	2	+0 -0	+98 -0	63 $\pm$ 1	+98 -0	49 $\pm$ 1
Blocks-SGPLAN	3	+12 -0	+70 -14	34 $\pm$ 14	+60 -26	9 $\pm$ 3
Gripper-FF	2	+0 -0	+100 -0	97 $\pm$ 0	+0 -100	-31 $\pm$ 0
Gripper-LPG	2	+0 -0	+100 -0	90 $\pm$ 0	+0 -100	-22 $\pm$ 0
Gripper-VHPOP	2	+34 -0	+66 -0	99 $\pm$ 0	+0 -56	-19 $\pm$ 1
Ferry-FF	2	+0 -6	+94 -0	76 $\pm$ 1	+0 -94	-70 $\pm$ 3
Ferry-LPG	2	+0 -0	+86 -0	89 $\pm$ 0	+0 -86	-21 $\pm$ 0
Ferry-SATPLAN	2	+16 -0	+84 -0	93 $\pm$ 2	+0 -84	-63 $\pm$ 4
Ferry-VHPOP	3	+76 -0	+22 -0	100 $\pm$ 0	+0 -22	-25 $\pm$ 3
NFerry-FF	2	+16 -4	+70 -0	54 $\pm$ 1	+0 -70	-5 $\pm$ 0
NFerry-SGPLAN	3	+16 -0	+52 -32	24 $\pm$ 13	+2 -82	-18 $\pm$ 1
Satellite-FF	2	+2 -34	+34 -0	95 $\pm$ 0	+0 -34	-45 $\pm$ 1
Satellite-VHPOP	4	+50 -0	+28 -0	91 $\pm$ 4	+0 -20	-14 $\pm$ 3
NSatellite-FF	2	+4 -8	+30 -0	99 $\pm$ 0	+4 -26	-8 $\pm$ 1
NSatellite-LPG	3	+62 -0	+42 -0	50 $\pm$ 4	+0 -36	-13 $\pm$ 1
NSatellite-SGPLAN	2	+0 -0	+100 -0	37 $\pm$ 0	+0 -100	-28 $\pm$ 1

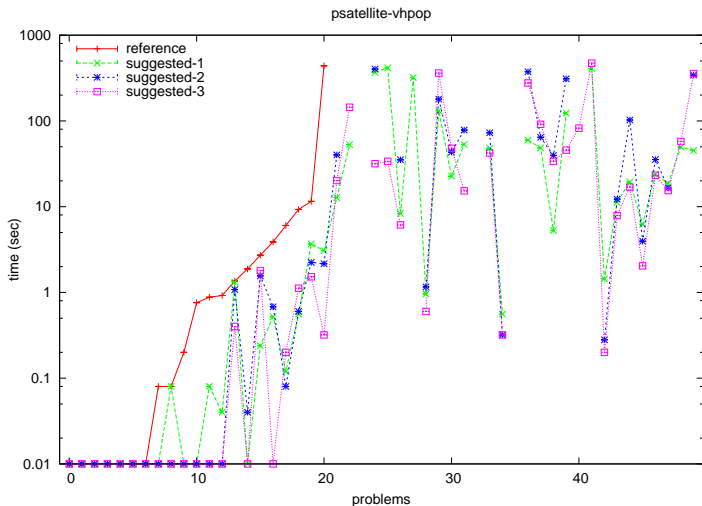
## Results: Summary Table Length Gain +L, Loss -l probs

<i>domain-planner</i>	N	+S -s	+T -t	P $\pm$ p	+L -l	Q $\pm$ q
Blocks-FF	3	+40 -0	+56 -0	92 $\pm$ 3	+52 -6	22 $\pm$ 2
Blocks-LPG	2	+0 -0	+98 -0	63 $\pm$ 1	+98 -0	49 $\pm$ 1
Blocks-SGPLAN	3	+12 -0	+70 -14	34 $\pm$ 14	+60 -26	9 $\pm$ 3
Gripper-FF	2	+0 -0	+100 -0	97 $\pm$ 0	+0 -100	-31 $\pm$ 0
Gripper-LPG	2	+0 -0	+100 -0	90 $\pm$ 0	+0 -100	-22 $\pm$ 0
Gripper-VHPOP	2	+34 -0	+66 -0	99 $\pm$ 0	+0 -56	-19 $\pm$ 1
Ferry-FF	2	+0 -6	+94 -0	76 $\pm$ 1	+0 -94	-70 $\pm$ 3
Ferry-LPG	2	+0 -0	+86 -0	89 $\pm$ 0	+0 -86	-21 $\pm$ 0
Ferry-SATPLAN	2	+16 -0	+84 -0	93 $\pm$ 2	+0 -84	-63 $\pm$ 4
Ferry-VHPOP	3	+76 -0	+22 -0	100 $\pm$ 0	+0 -22	-25 $\pm$ 3
NFerry-FF	2	+16 -4	+70 -0	54 $\pm$ 1	+0 -70	-5 $\pm$ 0
NFerry-SGPLAN	3	+16 -0	+52 -32	24 $\pm$ 13	+2 -82	-18 $\pm$ 1
Satellite-FF	2	+2 -34	+34 -0	95 $\pm$ 0	+0 -34	-45 $\pm$ 1
Satellite-VHPOP	4	+50 -0	+28 -0	91 $\pm$ 4	+0 -20	-14 $\pm$ 3
NSatellite-FF	2	+4 -8	+30 -0	99 $\pm$ 0	+4 -26	-8 $\pm$ 1
NSatellite-LPG	3	+62 -0	+42 -0	50 $\pm$ 4	+0 -36	-13 $\pm$ 1
NSatellite-SGPLAN	2	+0 -0	+100 -0	37 $\pm$ 0	+0 -100	-28 $\pm$ 1

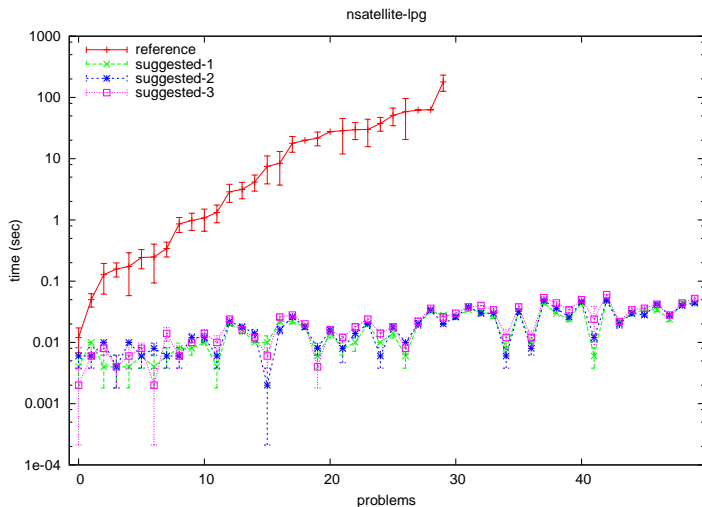
Results: Summary Table Length Gain Mean(Q) $\pm$ Disp(q)

<i>domain-planner</i>	N	+S -s	+T -t	P $\pm$ p	+L -l	Q $\pm$ q
Blocks-FF	3	+40 -0	+56 -0	92 $\pm$ 3	+52 -6	22 $\pm$ 2
Blocks-LPG	2	+0 -0	+98 -0	63 $\pm$ 1	+98 -0	49 $\pm$ 1
Blocks-SGPLAN	3	+12 -0	+70 -14	34 $\pm$ 14	+60 -26	9 $\pm$ 3
Gripper-FF	2	+0 -0	+100 -0	97 $\pm$ 0	+0 -100	-31 $\pm$ 0
Gripper-LPG	2	+0 -0	+100 -0	90 $\pm$ 0	+0 -100	-22 $\pm$ 0
Gripper-VHPOP	2	+34 -0	+66 -0	99 $\pm$ 0	+0 -56	-19 $\pm$ 1
Ferry-FF	2	+0 -6	+94 -0	76 $\pm$ 1	+0 -94	-70 $\pm$ 3
Ferry-LPG	2	+0 -0	+86 -0	89 $\pm$ 0	+0 -86	-21 $\pm$ 0
Ferry-SATPLAN	2	+16 -0	+84 -0	93 $\pm$ 2	+0 -84	-63 $\pm$ 4
Ferry-VHPOP	3	+76 -0	+22 -0	100 $\pm$ 0	+0 -22	-25 $\pm$ 3
NFerry-FF	2	+16 -4	+70 -0	54 $\pm$ 1	+0 -70	-5 $\pm$ 0
NFerry-SGPLAN	3	+16 -0	+52 -32	24 $\pm$ 13	+2 -82	-18 $\pm$ 1
Satellite-FF	2	+2 -34	+34 -0	95 $\pm$ 0	+0 -34	-45 $\pm$ 1
Satellite-VHPOP	4	+50 -0	+28 -0	91 $\pm$ 4	+0 -20	-14 $\pm$ 3
NSatellite-FF	2	+4 -8	+30 -0	99 $\pm$ 0	+4 -26	-8 $\pm$ 1
NSatellite-LPG	3	+62 -0	+42 -0	50 $\pm$ 4	+0 -36	-13 $\pm$ 1
NSatellite-SGPLAN	2	+0 -0	+100 -0	37 $\pm$ 0	+0 -100	-28 $\pm$ 1

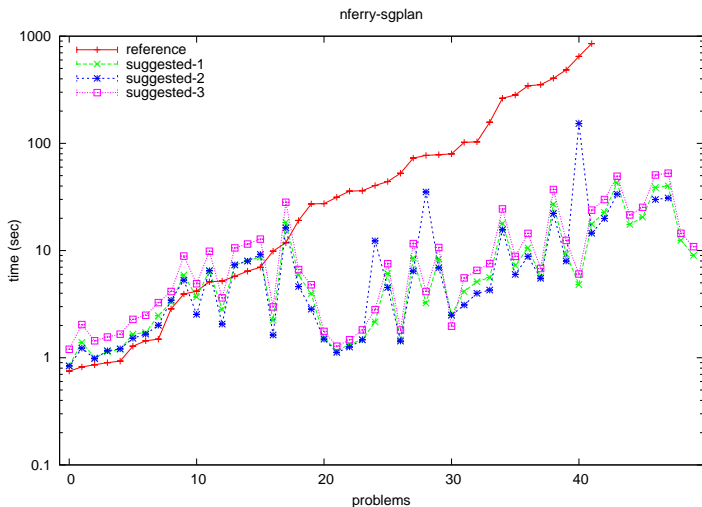
# Results: PSatellite VHPOP



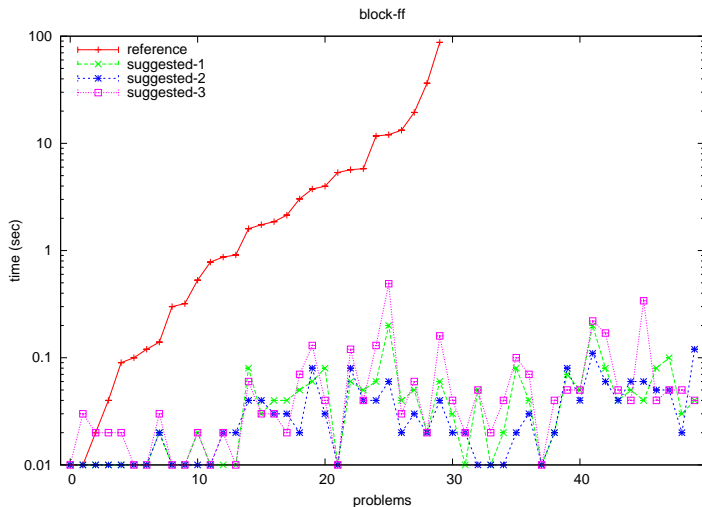
## Results: NSatellite LPG



## Results: NFerry SGPlan



# Results: Blocks Fast-Forward



# Analysis

## Hypotheses

1. Non-singleton bunches are more likely.
2. Bunches are not like to have consecutive top chunks.
3. Both of the above hypotheses seem to hold positively.
4. Complex domains have to be tried, if planners can cope with.

## Training Time: parameters to be tuned

hours	fd	ff	lpg	satplan	sgplan	vhpop
blocks	14.5	19.5	58.7	3.1	9.5	0.5
ferry	17.2	16.9	34.2	1.1	19.1	0.5
gripper	6.6	3.9	22.3	1.1	3.0	0.9
satellite	31.2	30.6	152.4	0.8	13.7	1.6
nferry		24.7			37.9	
nsatellite		46.5	37.2		28.3	

CPU 3GHZ  
RAM 2GB  
RankTime 10s  
RankMem 1GB  
EpochLim 200  
PopSize 2actns  
Replc 1/25epch

# Analysis

## Hypotheses

1. Non-singleton bunches are more likely.
2. Bunches are not like to have consecutive top chunks.
3. Both of the above hypotheses seem to hold positively.
4. Complex domains have to be tried, if planners can cope with.

## Training Time: parameters to be tuned

hours	fd	ff	lpg	satplan	sgplan	vhpop	CPU 3GHZ
blocks	14.5	19.5	58.7	3.1	9.5	0.5	RAM 2GB
ferry	17.2	16.9	34.2	1.1	19.1	0.5	RankTime 10s
gripper	6.6	3.9	22.3	1.1	3.0	0.9	RankMem 1GB
satellite	31.2	30.6	152.4	0.8	13.7	1.6	EpochLim 200
nferry		24.7			37.9		PopSize 2actns
nsatellite		46.5	37.2		28.3		Replc 1/25epch

# Related Work

## Key Differences

- ▶ Existing methods elect 2 top macros without considering interactions.
- ▶ Unlike existing methods, this method deals with acquisition part only.
- ▶ Existing methods have no search guidance on macro/combination space.

Characteristic	Domain/Planner	Solution
Plateau escaping sequence	Fast Forward	MARVIN
Valley travelling sequence	A heu. planner	MACLEARN
Causally linked macro	Fast Forward	Macro-FF
Don't care; don't know	Any planner	Wizard
Symmetry, almost symmetry	Briefcase, etc.	MARVIN
Component abstraction	Rover, etc.	Macro-FF
Operator decomposability	8 Puzzle, etc.	Korf's MPS
Don't care; don't know	Any domain	Wizard

# Contribution

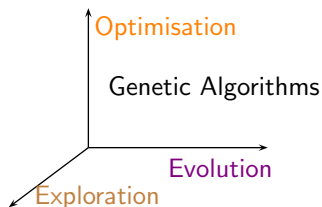
## Most Existing Methods

- ▶ Acquire macros from specific domain or planner properties
- ▶ Suggest a number of top performing macros learnt
- ▶ Are actually planners with limited macro learning capability

## Our Method

- ▶ Acquires best macro-combinations 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

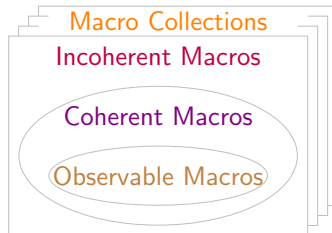
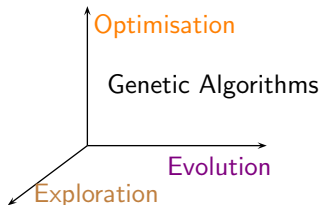
# Our Further Work



## 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.

## 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)

