

A Mean-Based Approach to Real-Time Planning

D. Pellier, B. Bouzy, M. Métivier
Damien.Pellier@paridescartes.fr
<http://www.math-info.univ-paris5.fr/~pellier/>

Laboratoire d'Informatique de Paris Descartes
Paris Descartes University



June 1st 2010

Plan

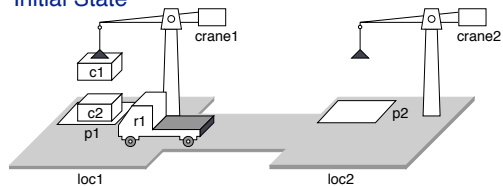
- ① Introduction
- ② Real-Time Search (RTS)
- ③ Upper Confidence for Tree (UCT)
- ④ Mean-based Heuristic Search for Planning (MHSP)
- ⑤ Experiments and Results
- ⑥ Discussion and conclusion

What is classical planning ?

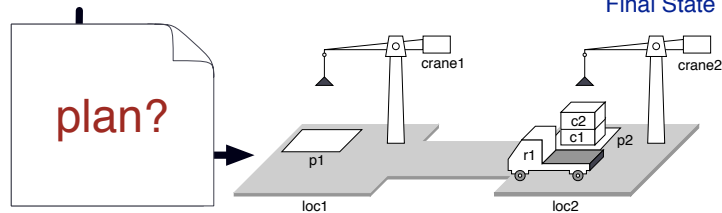
- An important domain of the Artificial Intelligence
- An abstract reasoning about actions that
 - takes as input
 - ① a description of actions
 - ② a description of the world
 - ③ some objective or goal
 - and returns a plan
 - a collection of actions whose global effect achieves the objective
- Find the whole plan before acting
- Use an action description language

Example: the docker robots

Initial State

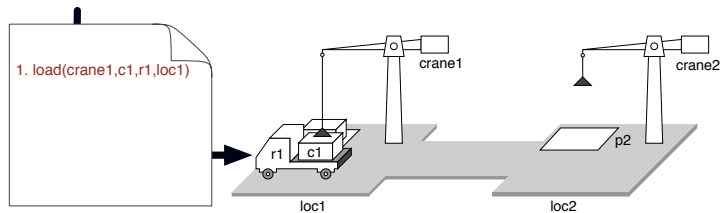
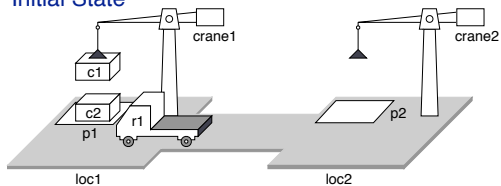


Final State



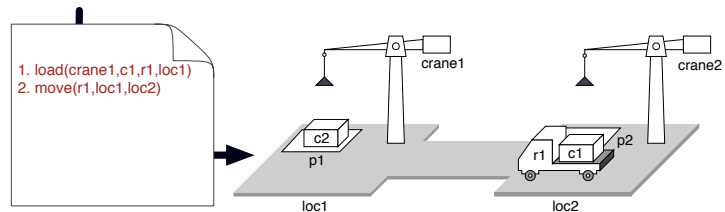
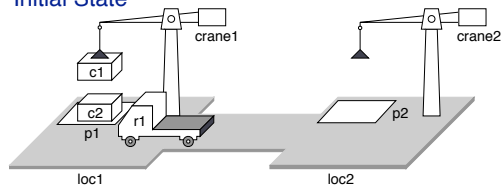
Example: the docker robots

Initial State



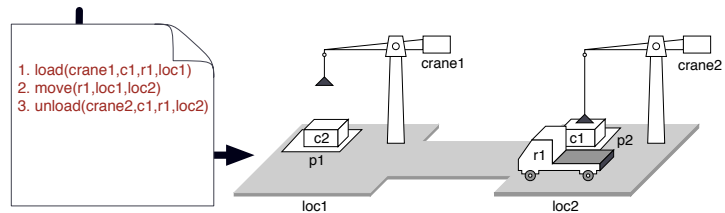
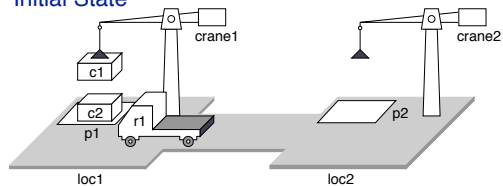
Example: the docker robots

Initial State



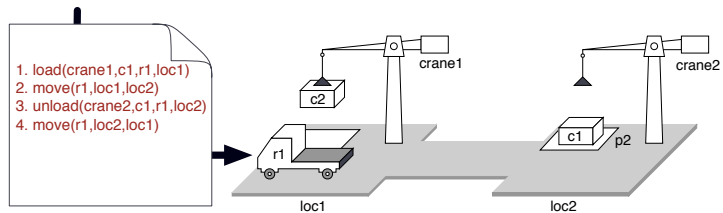
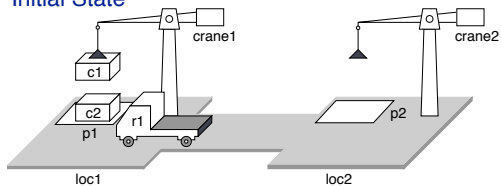
Example: the docker robots

Initial State



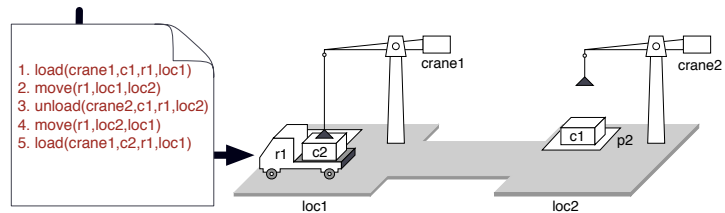
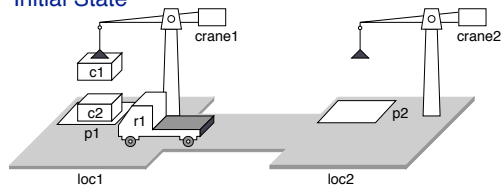
Example: the docker robots

Initial State



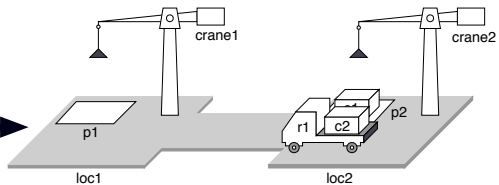
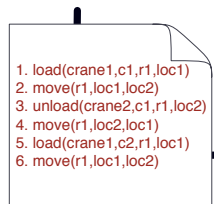
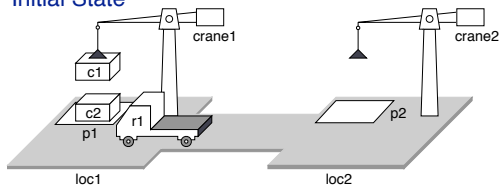
Example: the docker robots

Initial State



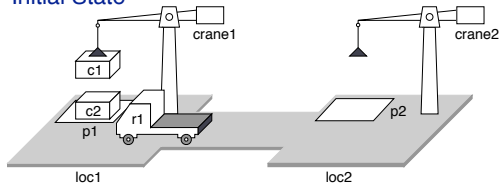
Example: the docker robots

Initial State



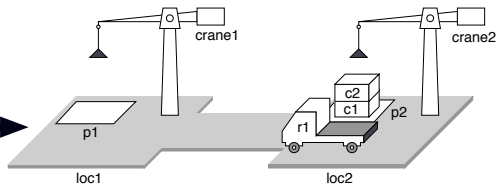
Example: the docker robots

Initial State



Final State

1. load(crane1,c1,r1,loc1)
2. move(r1,loc1,loc2)
3. unload(crane2,c1,r1,loc2)
4. move(r1,loc2,loc1)
5. load(crane1,c2,r1,loc1)
6. move(r1,loc1,loc2)
7. unload(crane2,c2,r1,loc2)



Example: the docker robots

- **Initial state** is described using logical language

$$s_0 = \{at(c1, loc1), at(c2, loc1), at(r1, loc1), \dots\}$$

- **The planning operators** describe the skills of the agent

:: A docker loads a container ?c in truck ?t at ?l

load(?l, ?c, ?t)

precond: at(?t, ?l), at(?c, ?l)

effects: $\neg at(?c, ?l)$, in(?t, ?c)

:: A docker unloads a container ?c from truck ?t at ?l

unload(?l, ?c, ?t)

precond: at(?t, ?l), in(?t, ?c)

effects: $\neg in(?t, ?c)$, at(?c, ?l)

What is anytime planning ?

- Anytime Planning
 - Find out a solution plan
 - Refine it to obtain an optimal plan
- Anytime strictly speaking
 - Provide a “something” at anytime
 - Refine it to obtain an optimal plan
- Something =
 - the first action
 - the first 2 actions
 - a partial plan
 - a solution plan
 - an optimal plan

Real-Time Search

- While not convergence, perform episodes
 - the planning problem is performed repeatedly
- 1 episode: while not goal, select an action, execute the action
- Real-time search: select an action in constant time ++
- Related works
 - Real-Time Heuristic Search, LRTA* (Korf 1990)
 - SLA* (Search and Learn A*) (Shue 1993)
 - FALCONS (Furcy 2000)
 - Weighted A* (Shimbo 2003)
 - LRTS γ -trap (Bulitko 2006)

Real-Time Search, Learning and Action Selection

- Learning or not: update the heuristic value or not

$$H(s) = \max(H(s), 1 + \min_i(H(s_i)))$$

- s : current state
- s_i : state of child i
- $H(s)$: Heuristic value of state s
- Action selection =
 - depth-one search for convergence proof in papers (Korf 1990) - -
 - real tree search for efficient planning agents: not well described - -
- With or without learning ?
 - with: convergence over episodes with depth-one action selection
 - without: planning depends on action selection efficiency

UCT and Computer Go

(Kocsis and Szepesvari, 2006)



- Computer Go
 - * – 2002: the classical period
 - 2003 – 2005: the basic Monte-Carlo period
 - 2006 – *: the UCT and the MCTS (Monte-Carlo Tree Search) period
 - Mogo, Crazy Stone, Fuego, Zen, etc.
- 2-player tournaments: time constraints
 - one hour per game
 - few seconds per move
- UCT is any-time strictly speaking

UCT for Computer Go

- While has time do
 - ① Browse the tree from the root to a leaf with the UCB rule
 - ② Expand the leaf with one node
 - ③ Perform a random game and get the result r
 - ④ Update the mean values of the browsed nodes with r
- The solution node is the node with the highest
 - number of visits
 - mean value

Upper Confidence Bound

(Auer and *al.* 2002)

- Choose the child i with the best upper confidence bound

$$N_{\text{select}} = \operatorname{argmax}_i (m_i + C \times \sqrt{\log(p)/n_i})$$

- m_i : mean value of choosing the child i
- n_i : number of times the child i was chosen
- p : number of times the parent node was chosen
- C : constant value set up experimentally to customize exploration

Mean-based Heuristic Search for real-time Planning (MHSP)

- Apply UCT on real-time planning for action selection
- MHSP = UCT + 2 adaptations:
 - ① Replace the random simulations by calls to a heuristic function
 - ② Initialize the mean value with heuristic values
- Planning assumption:
 - implicate time, deterministic effects, completely observable

MHSP

Replace the simulations by a heuristic function

- In 2-player games
 - The random simulations complete
 - Every simulations get a return
 - Evaluation functions can be hard to model, or compute
- In planning problems
 - The random simulations complete very rarely
 - Heuristic functions are efficiently used by state-of-the-art planners

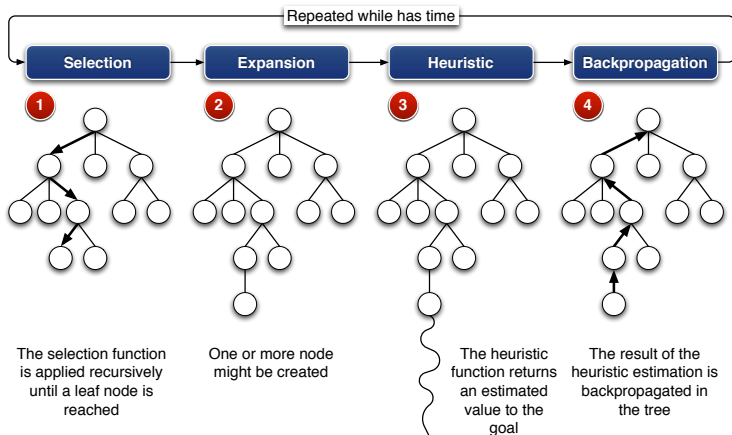
MHSP

Initialize the mean values with optimistic values

- Exploration in 2-player games
 - Do not use the $C \times \sqrt{\log(p)/n_i}$ term
 - Use knowledge-based optimistic values to initialize the mean values
- In planning problems
 - Do not use the $C \times \sqrt{\log(p)/n_i}$ term
 - Use knowledge-based optimistic values to initialize the mean values
 - An admissible heuristic is also an optimistic value ++

MHSP

Algorithm



Experiments and Results

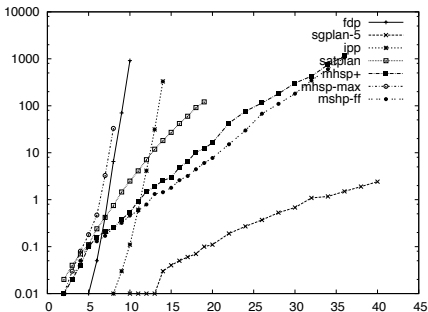
Preliminary state of the art comparison without the real-time constraint

- Observations

- On International Planning Competition (IPC) benchmarks, MHSP is inferior to the best state-of-the-art planners
- Without time constraints MHSP is surpassed
 - ↳ MHSP and UCT are anytime algorithms
 - ↳ The real-time constraint is required to make value of UCT

- Example on the blockworld problem:

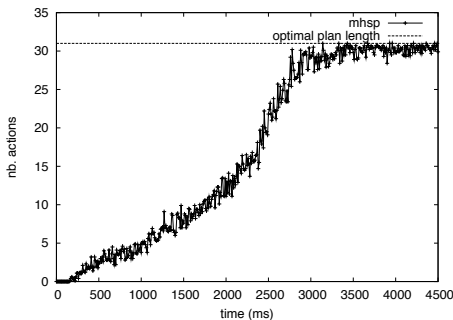
- x = size of the problem
- y (log scale) = time (ms)



Experiments and Results

However...

- Example on a blockworld problem (17 blocks) :
 - x = time (ms)
 - y = number of actions in the optimal plan found by MHSP



Experiments and Results

Settings

- Decision time parameter for action selection
 - from 40 ms to 4,000 ms
- Other algorithms for action selection
 - adapted A*
 - Breadth-First Search (BFS) (similar to γ -trap)
- Three tests
 - test 1: gives the plan length in the decision time
 - test 2: test 1 + learning
 - test 3: assess the quality of plans with distances
- Four domains
 - blocksworld
 - ferry
 - gripper
 - satellite

Experiments and Results

Test 1 and 2, Ferry

		No Learning			Learning		
problem	search algo.	avr. time	min lgth.	fail %	avr. time	min lgth.	fail %
ferry-05 time: 40ms opt p lgth: 18	A*	1.09	19	0	0.68	18	0
	BFS	8.91	18	16	12.03	18	14
	MHSP	0.59	18	0	0	18	0
ferry-10 time: 200ms opt p lgth: 35	A*	97.9	42	32	9.36	37	0
	BFS	9.05	35	0	17.22	35	0
	MHSP	6.26	35	0	6.24	35	0
ferry-15 time: 2000ms opt p lgth: 51	A*	157.85	58	55	112.03	57	55
	BFS	103.75	51	0	104.31	51	0
	MHSP	86.45	51	0	87.39	51	0
ferry-20 time: 4000ms opt p lgth: 73	A*	–	–	100	247.32	107	0
	BFS	–	–	100	284.44	75	35
	MHSP	260.49	73	0	255.31	73	15

Experiments and Results

Test 1 and 2, Gripper

		No Learning			Learning		
problem	search algo.	avr. time	min lgth.	fail %	avr. time	min lgth.	fail %
gripper-05 time: 50ms opt p lgth: 15	A*	0.56	15	0	0.45	15	0
	BFS	0.71	15	0	35.72	15	68
	MHSP	0.49	15	0	0.48	15	0
gripper-10 time: 165ms opt p lgth: 29	A*	92.28	33	36	76.48	29	0
	BFS	7.86	37	0	9.96	29	0
	MHSP	5.08	29	0	4.8	29	0
gripper-15 time: 450ms opt p lgth: 45	A*	160.1	77	70	56.32	49	14
	BFS	31.42	45	0	36.92	45	4
	MHSP	38.53	53	0	36.52	45	0
gripper-20 time: 1100ms opt p lgth: 59	A*	–	–	100	–	–	100
	BFS	102.76	61	0	132.44	73	2
	MHSP	134.86	73	0	99.47	73	0

Experiments and Results

Test 1 and 2, Satellite

		No Learning			Learning		
problem	search algo.	avr. time	min lgth.	fail %	avr. time	min lgth.	fail %
satellite-05 time: 300ms opt p lgth: 15	A*	3.49	15	0	3.54	15	0
	BFS	20.28	19	0	6.1	19	0
	MHSP	3.01	15	0	3.17	15	0
satellite-10 time: 2000ms opt p lgth: 29	A*	–	–	100	–	–	100
	BFS	–	–	100	–	–	100
	MHSP	67.912	31	0	65.612	30	0

Experiments and Results

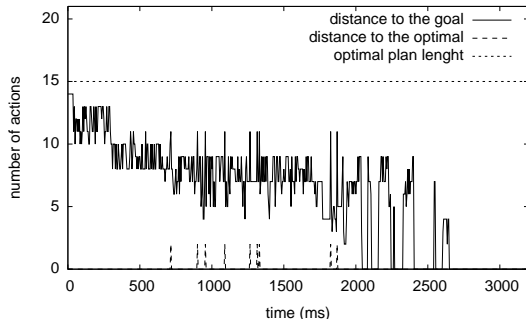
Partial plan distance definitions

- Useful for test 3 partial plan assessment
- Distance to goal (DG)
 - Optimal distance from the last state of the partial plan to the goal
 - Computed offline with A*
- Distance to optimum (DO)
 - Distance to goal + partial plan length - optimal plan length
 - Assess the deviation from the optimal plan
 - We look for partial plan with distance to optimum as small as possible
 - High computation cost
- For an optimal plan, $DG = DO = 0$
- When $DO = 0$, the partial plan is good (part of an optimal plan)
- When $DG = 0$, the partial plan is a solution plan

Experiments and Results

test 3 A*

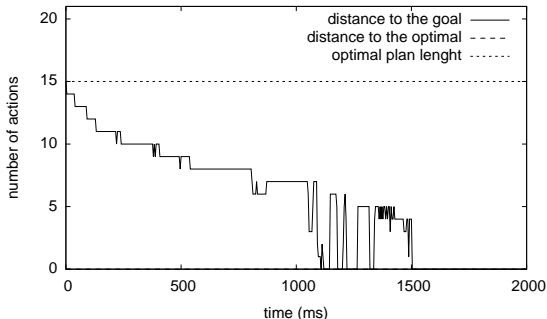
- Example on a gripper problem :
 - x (log scale) time (ms)
 - $y = DO$ and DG



Experiments and Results

test 3 BFS

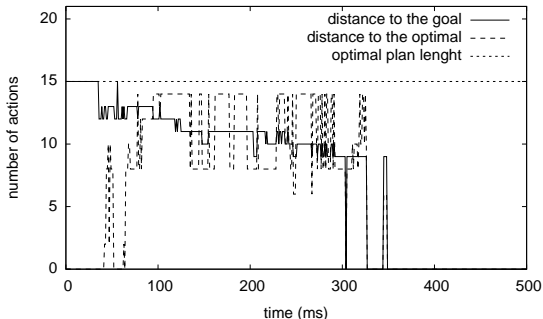
- Example on a gripper problem :
 - x (log scale) time (ms)
 - $y = DO$ and DG



Experiments and Results

test 3 MHSP

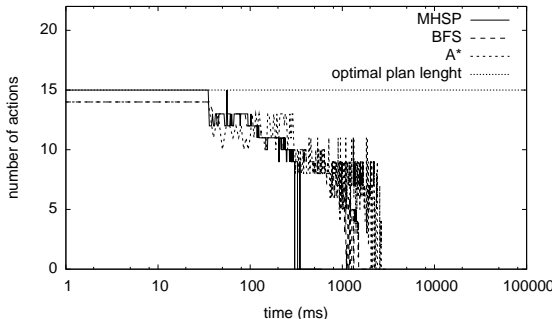
- Example on a gripper problem :
 - x (log scale) time (ms)
 - $y = DO$ and DG



Experiments and Results

Comparison with A* and BFS

- Example on a gripper problem :
 - x (log scale) time (ms)
 - $y = DO$ and DG



Discussion

- Related works
 - (Chaslot 2006) Production management problems
 - the simulations complete
 - the return depends on the quantity of product, can be computed at any time
 - (Nakhost 2009) Monte-Carlo Planning, Arvand
 - several simulations are launched
 - after each simulation, the heuristic value is computed
 - the best simulation is kept, and the corresponding sequence executed
 - very good results on some domains of IPC
 - (Bjarnasson 2009) Solitaire Klondike game

Conclusion

- To sum up
 - MHSP = Adaptation of UCT for planning
 - MHSP can be used for the action selection stage of Real-Time Search
 - MHSP needs the real-time constraint to be valuable
 - MHSP assessment biased by the existence of very good heuristic functions
- Future works
 - Experiments on classical planning problems are going on
 - Since partial plans are available, execute a sequence of actions (similar to γ -trap)
 - Reuse previous search tree from state to state
 - Adapt MHSP to non-deterministic planning problems