



# Generating Plans in Concurrent, Probabilistic, Oversubscribed Domains

---

Li Li and Nilufer Onder

Department of Computer Science

Michigan Technological University

(Presented by: Nilufer Onder)

ICAPS'07 3<sup>rd</sup> Workshop on Planning and Execution for Real World Systems

September 22, 2007



# Main features

---

- Aspects of complex domains
  - Deadlines, limited resources
  - Concurrency
  - Failures
  - Oversubscription
- Two types of parallel actions
  - Redundant (“early finish”)
  - Different goals (“all finish”)
- Aborting actions
  - When it succeeds
  - When it fails



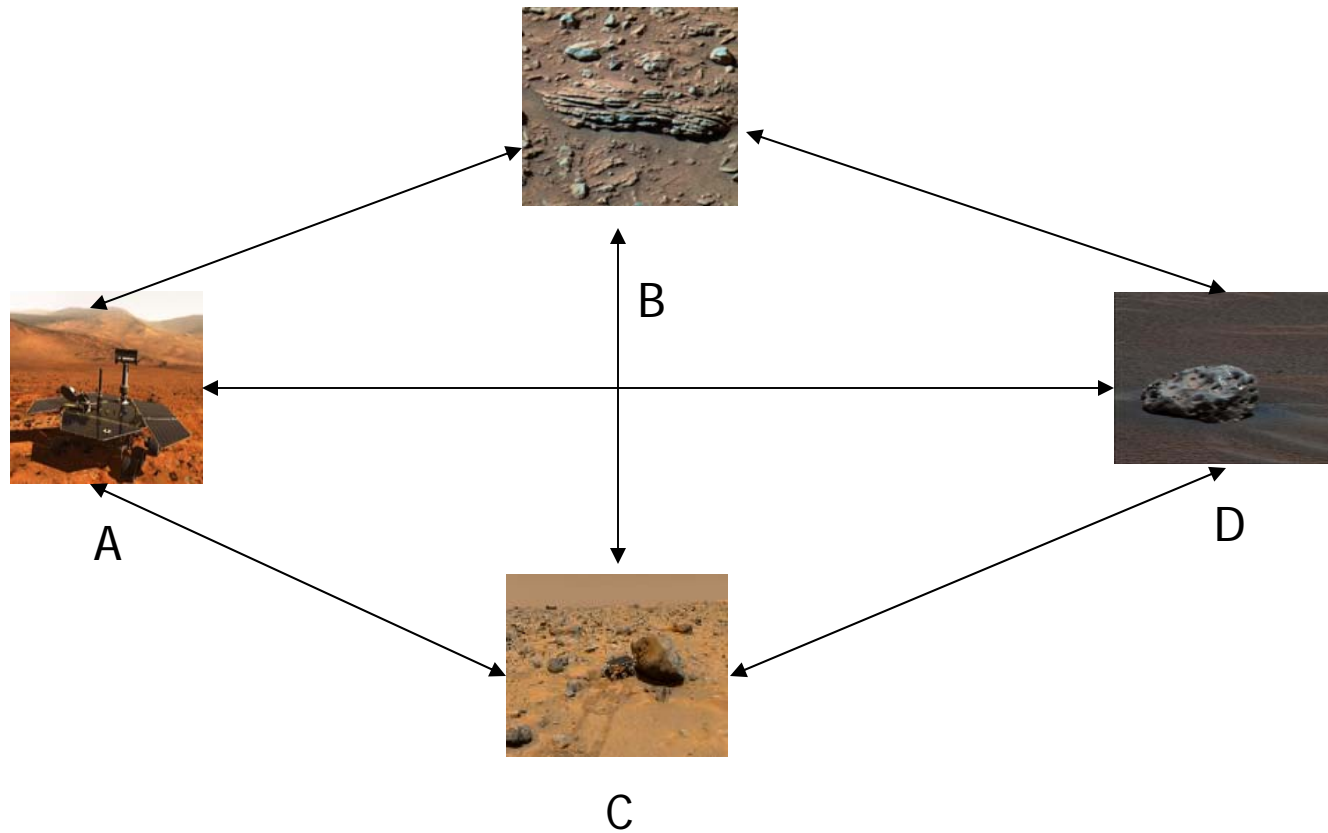
# Outline

---

- Example domain
- Dichotomy of concurrent actions
- AO\* and CPOAO\* algorithms
- Heuristics used in CPOAO\*
- Caching
- Conclusion and future work

# A simple Mars rover domain

Locations A, B, C and D on Mars:





## The actions

---

Action	Success probability	Description
Move(L1,L2)	100%	Move the rover from Location L1 to location L2
Sample (L)	70%	Collect a soil sample at location L
Camera (L)	60%	Take a picture at location L



# Problem 1

---

- Initial state:
  - The rover is at location A
  - No other goals have been achieved
- Rewards:
  - $r_1 = 10$ : Get back to location A
  - $r_2 = 2$ : Take a picture at location B
  - $r_3 = 1$ : Collect a soil sample at location B
  - $r_4 = 3$ : Take a picture at location C



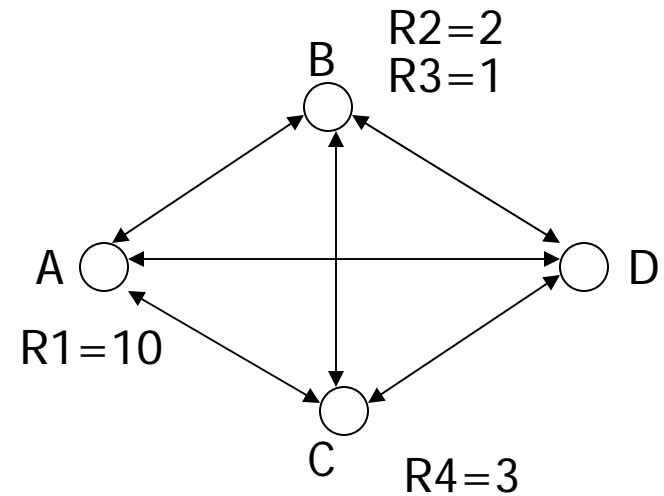
# Problem 1

---

- Time Limit:
  - The rover is only allowed to operate for 3 time units
- Actions:
  - Each action takes 1 time unit to finish
  - Actions can be executed in parallel if they are compatible

# A solution to problem 1

- (1) Move (A, B)
- (2) Camera (B)  
Sample (B)
- (3) Move (B, A)





## Add redundant actions

---

- Actions **Camera0** (60%) and **Camera1** (50%) can be executed concurrently.
- There are two rewards:
  - R1: Take a picture P1 at location A
  - R2: Take a picture P2 at location A



# Dichotomy of concurrent actions

---

- All finish: Speed up the execution  
Use concurrent actions to achieve different goals.
- Early finish: Redundancy for critical tasks  
Use concurrent actions to achieve the same goal.



## Example of all finish actions

---

- If  $R1=10$  and  $R2=10$ ,  
the best strategy is to execute Camera0 to  
achieve one reward and execute Camera1 to  
achieve another.  
(All finish)

$$\begin{aligned} \text{The expected total rewards} \\ &= 10 * 60\% + 10 * 50\% \\ &= 11 \end{aligned}$$



## Example of early finish actions

---

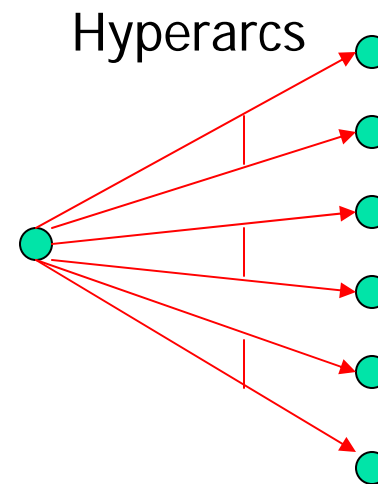
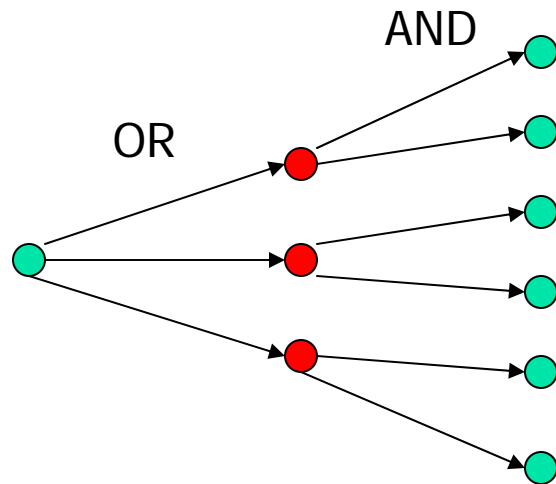
- If  $R1=100$  and  $R2=10$ ,  
the best strategy is to use both Camera0 and  
Camera1 to achieve R1.  
(Early finish)

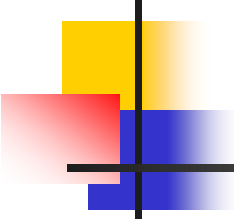
The expected total rewards

$$\begin{aligned} &= 100 * 50\% + (100 - 100 * 50\%) * 60\% \\ &= 50 + 30 = 80 \end{aligned}$$

# The AO\* algorithm

AO\* searches in an and-or graph (hypergraph)

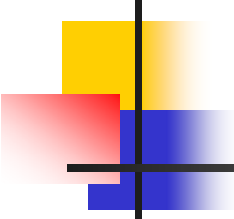




# Concurrent Probabilistic Over-subscription AO\* (CPOAO\*)

---

- Consider action combinations instead of individual actions.
- Use hyperarcs to represent action combinations.
- Extend the states to include the remaining resources
- Extend the states to include unfinished actions



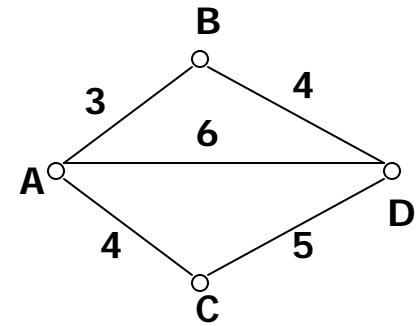
## Concurrent Probabilistic Over-subscription AO\* (CPOAO\*)

---

- Unfinished actions are put back into the set of applicable actions and can be aborted
- The heuristic function estimates the total of the achievable rewards for the newly generated nodes.

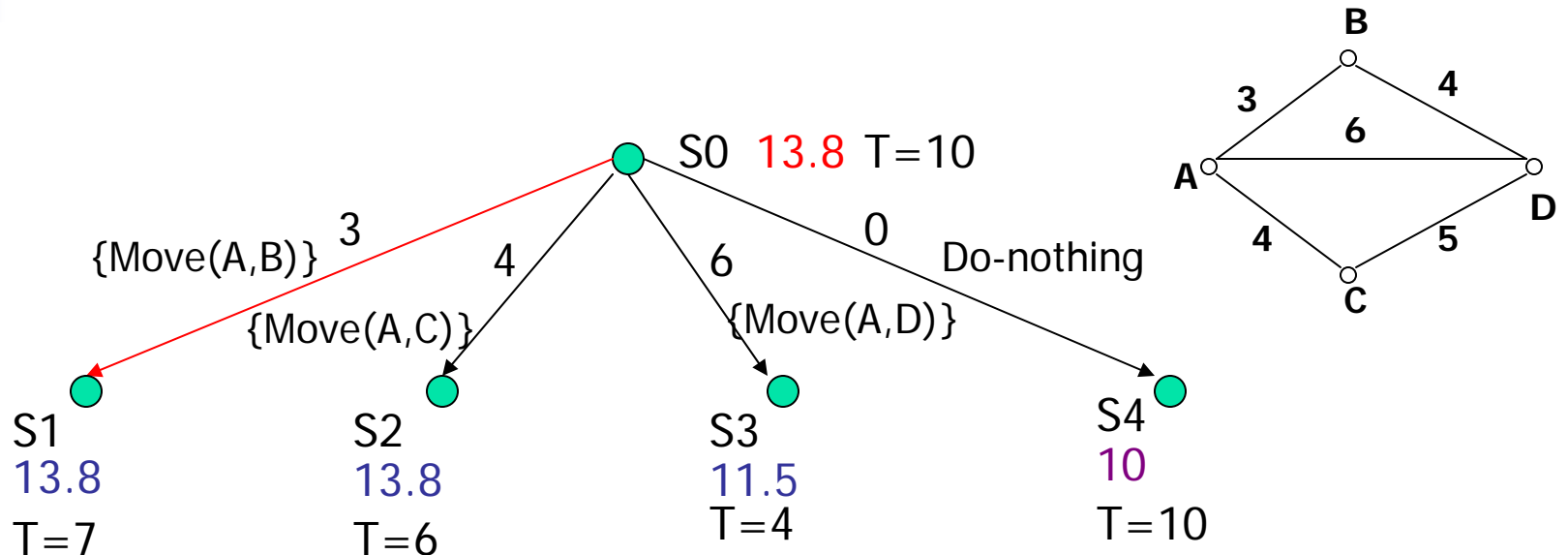
# CPOAO\* search

● S0  
14.66



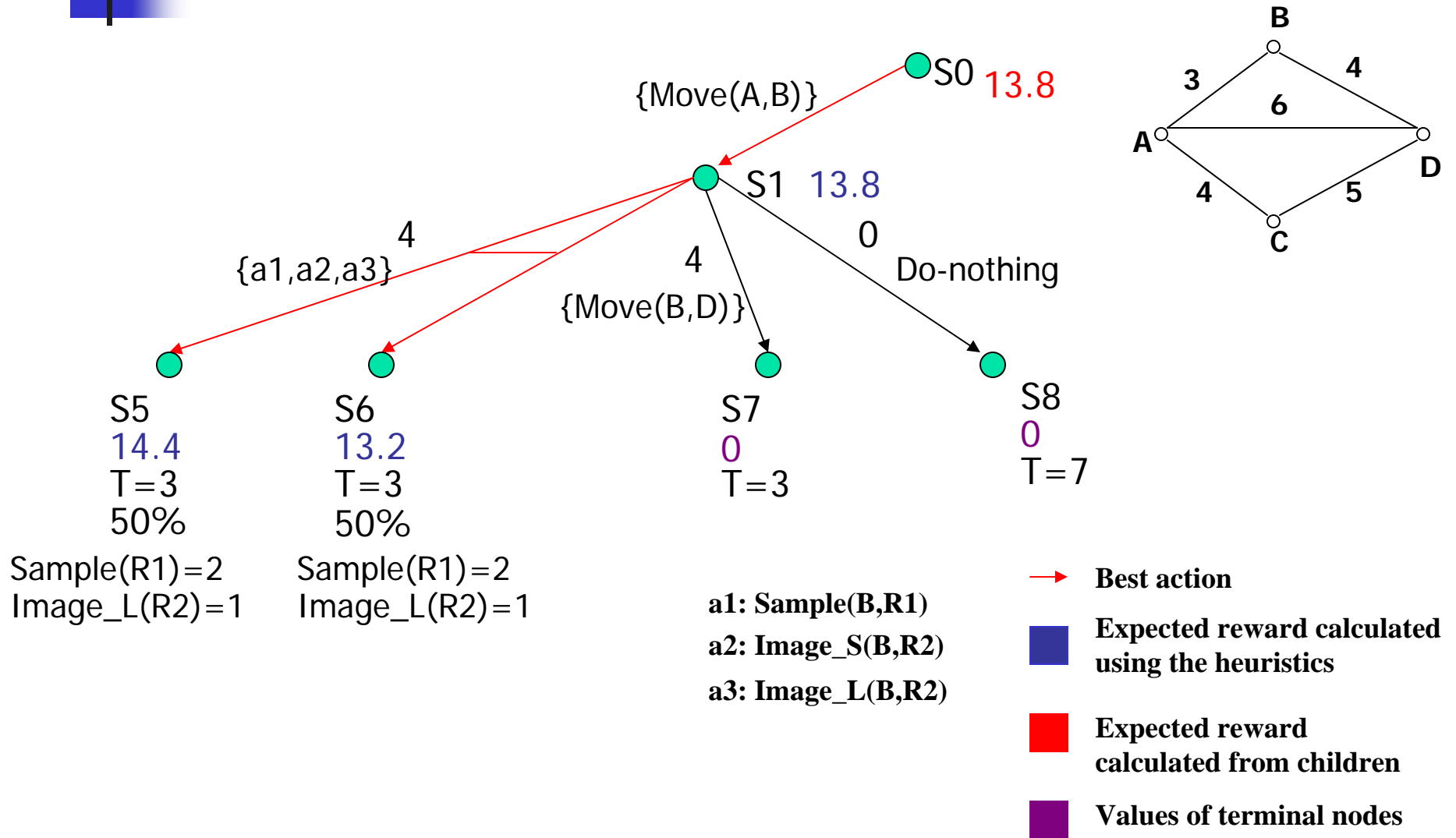
■ Expected reward calculated using the heuristics

# CPOAO\* Search

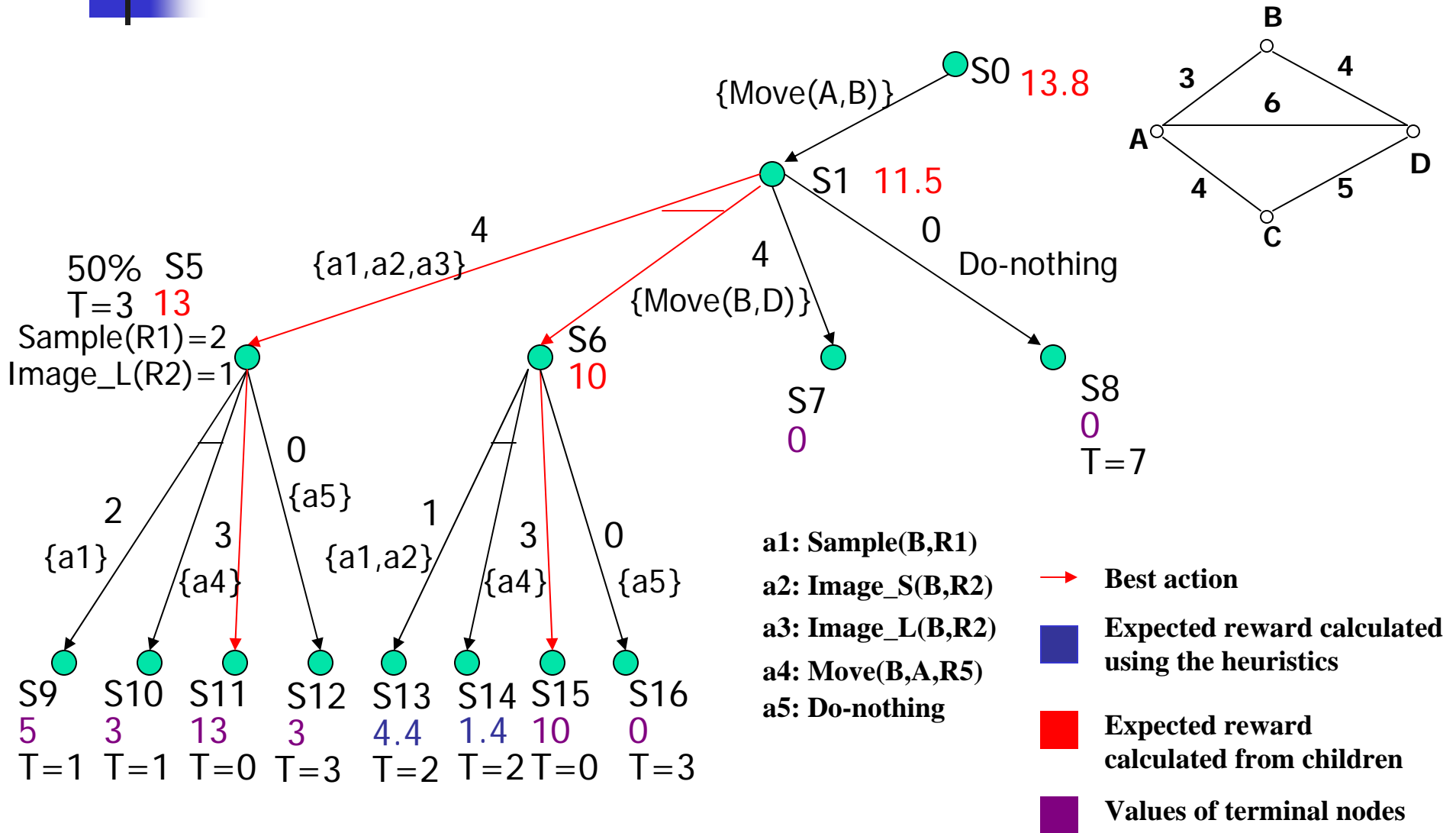


- Best action
- Expected reward calculated using the heuristics
- Expected reward calculated from children
- Values of terminal nodes

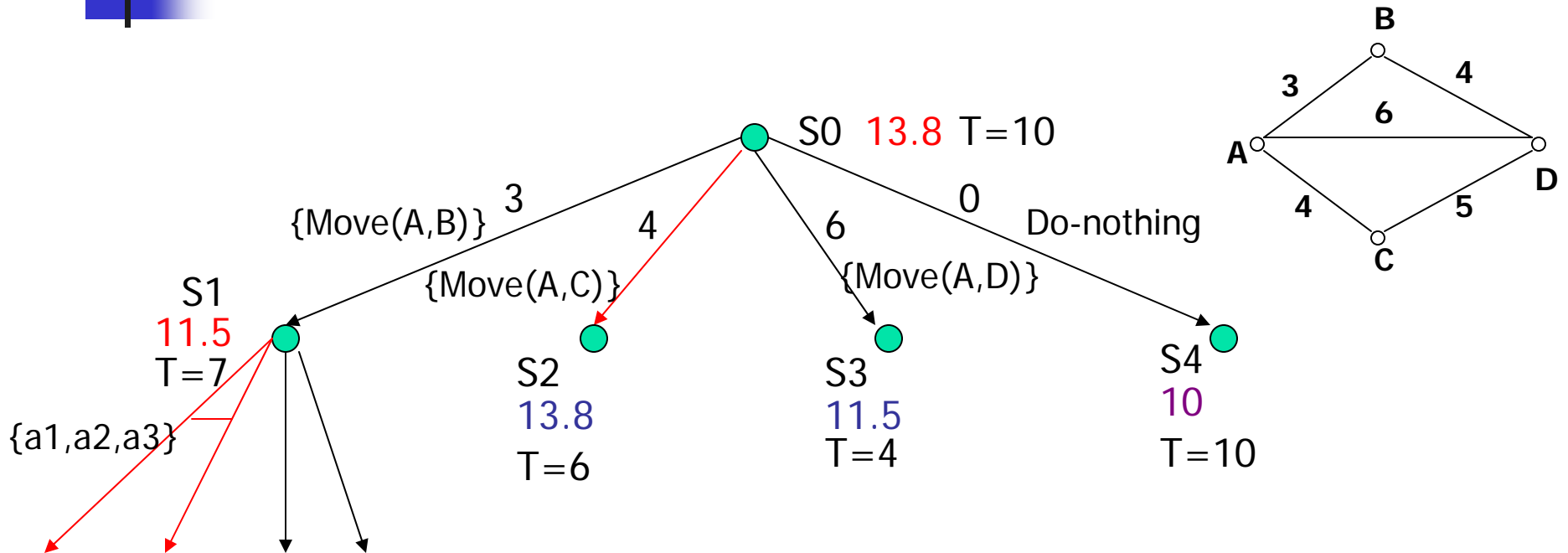
# CPOAO\* search



# CPOAO\* search



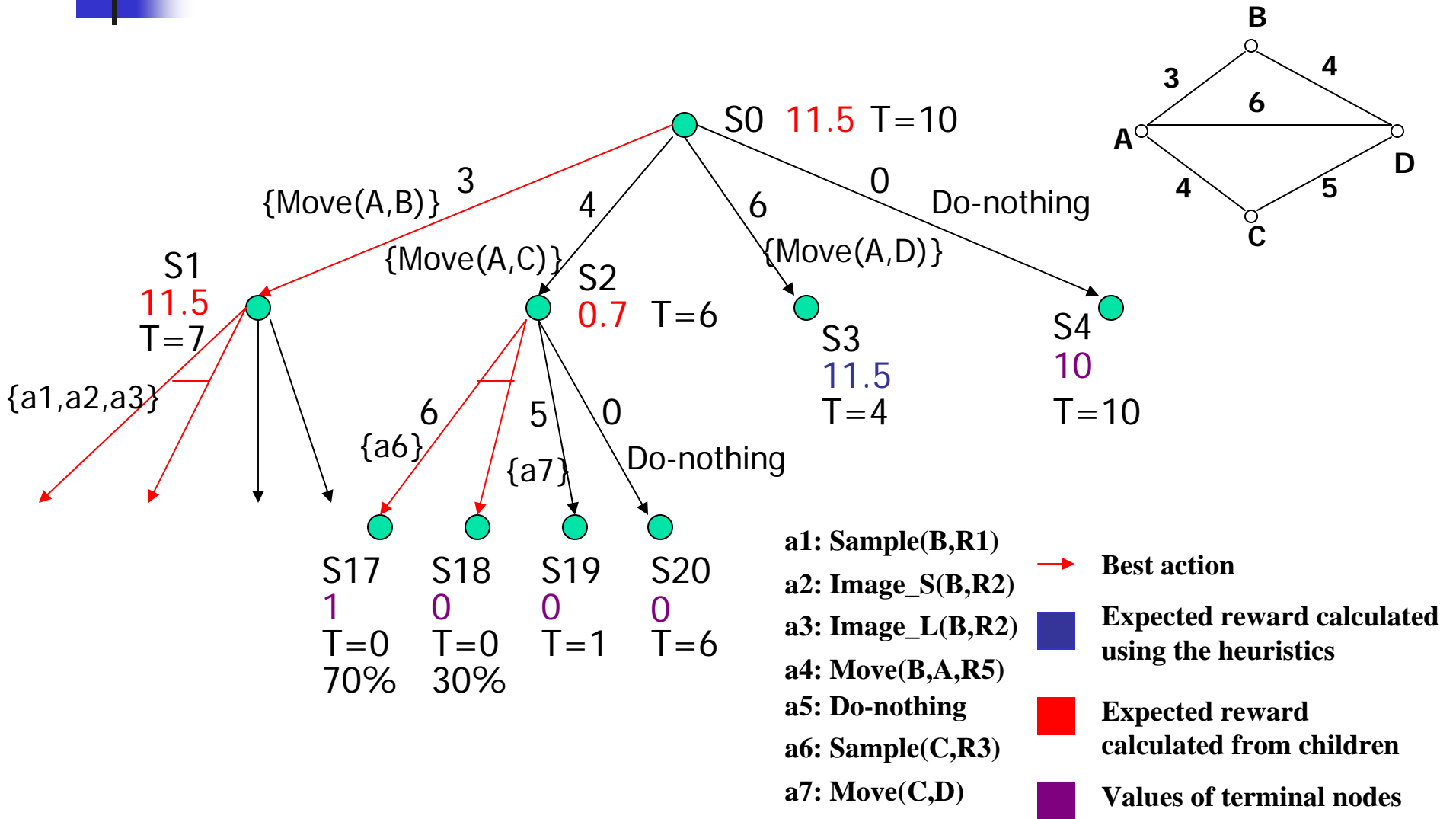
# CPOAO\* search



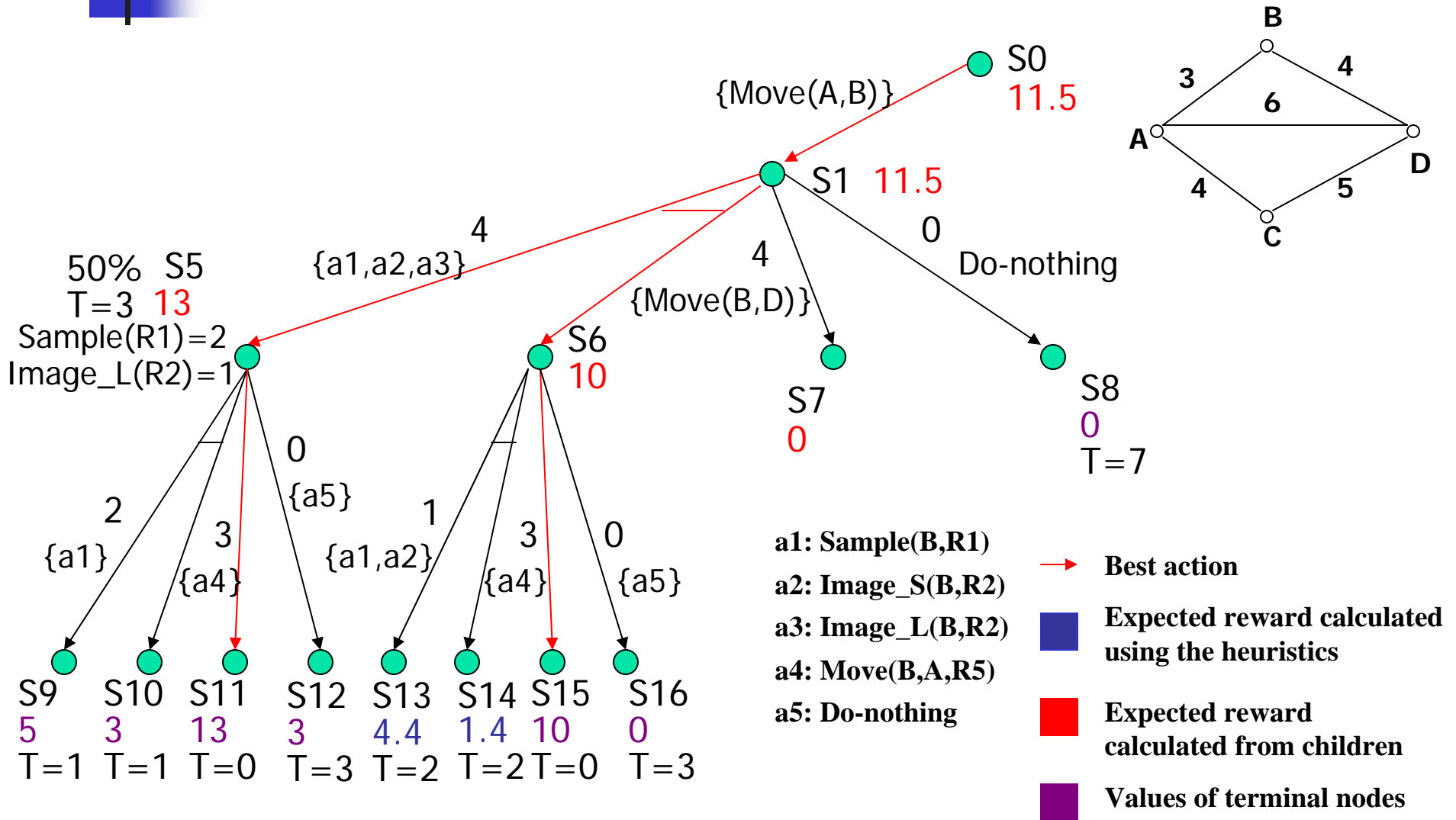
- a1: Sample(B,R1)
- a2: Image\_S(B,R2)
- a3: Image\_L(B,R2)
- a4: Move(B,A,R5)
- a5: Do-nothing

- Best action
- Expected reward calculated using the heuristics
- Expected reward calculated from children
- Values of terminal nodes

# CPOAO\* search



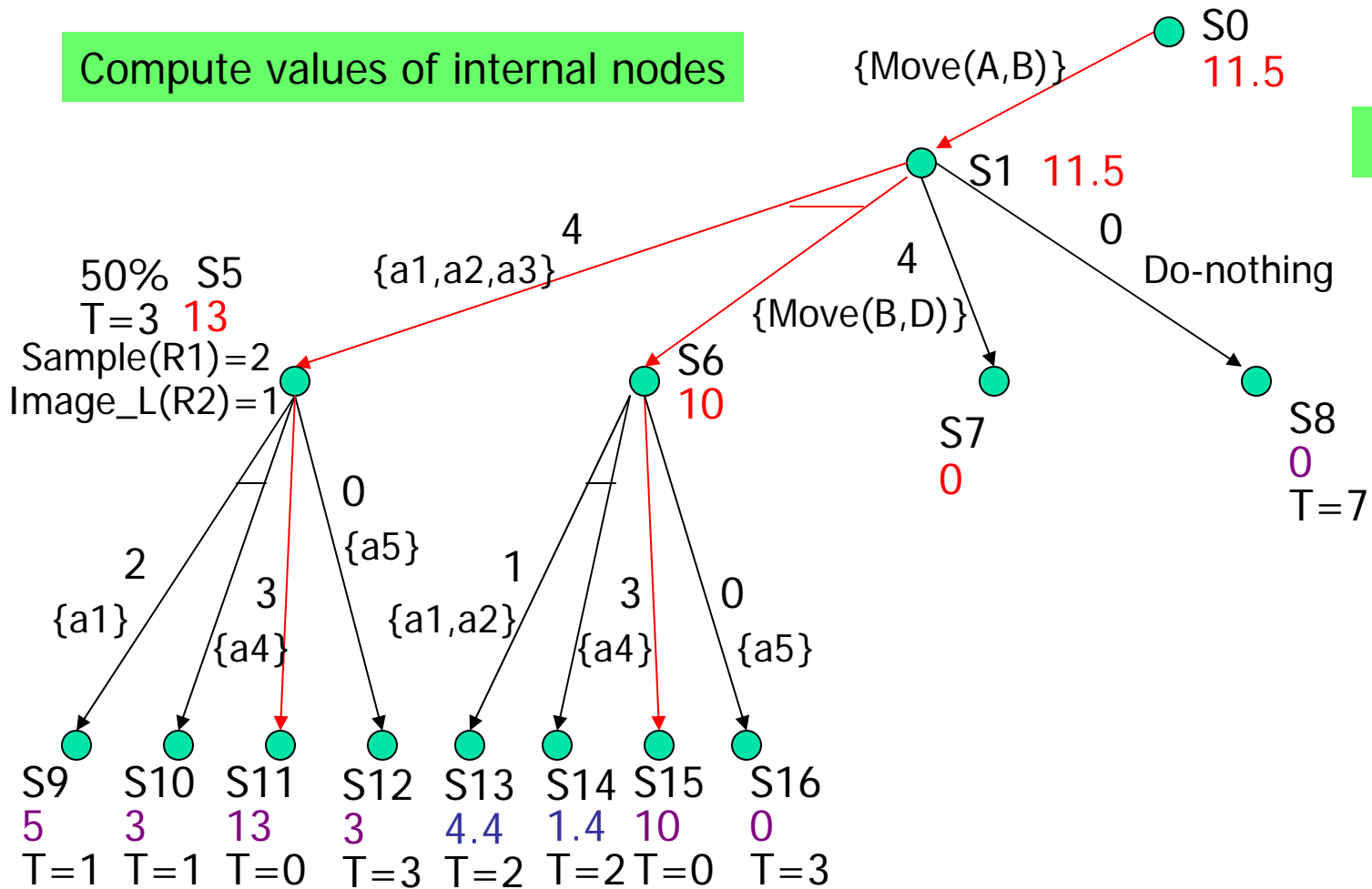
# CPOAO\* search



# CPOAO\* search improvements

Compute values of internal nodes

Prune children





## Heuristics used by CPOAO\*

---

- A simple heuristic function is used to estimate the total expected reward for the newly generated states.
- A second heuristic function is used to prune the branches of the concurrent action sets when time is the only resource. The idea is to remove the branches which will not be part of the solution plan.



## Caching in CPOAO\*

---

- In the search space, there are many similar states which differentiate only on the remaining resources.
- We can use the values of the states which have more remaining resources as the upper bound for the values of the states which have less resources.
- The result is very promising. Up to 45% of the total states are pruned.



# Experimental work

---

- Mars rover problem with 4 actions
  - Move
  - Collect-Soil-Sample
  - Camera0
  - Camera1
- 3 sets of experiments
  - Gradually increase complexity
  - Ablation studies
  - Cache vs. no-cache



# Results of complexity experiments

Problem	TR	T = 20			T = 40		
		ER	NG	ET	ER	NG	ET
5-5-5	34	22.4	41	< 1	23.84	524	< 1
5-5-10	51	22.4	95	< 1	29.6	7407	3
5-10-5	34	22.6	178	< 1	26.08	11374	6
5-10-10	51	25.6	389	< 1	32.96	87858	155
10-10-8	41	23.8	69	< 1	25.89	1130	< 1
10-10-21	84	25.3	265	< 1	30.99	27538	19
10-17-8	41	22.4	122	< 1	25.6	8179	2
10-17-21	84	23.9	332	< 1	28.9	102484	190
12-12-12	52	23.8	124	< 1	27.52	3625	1
12-12-23	97	25.3	301	< 1	30.14	36344	32
12-21-12	52	22.4	196	< 1	25.6	16724	8
12-21-23	97	23.9	396	< 1	28.9	125959	272
15-16-14	58	23.8	184	< 1	27.52	8664	3
15-16-31	116	25.3	450	< 1	-	-	-
15-28-14	58	22.4	337	< 1	27.4	51344	75
15-28-31	116	25.3	752	< 1	-	-	-

**Problem:** locations-  
paths-rewards

**TR:** Sum of all rewards

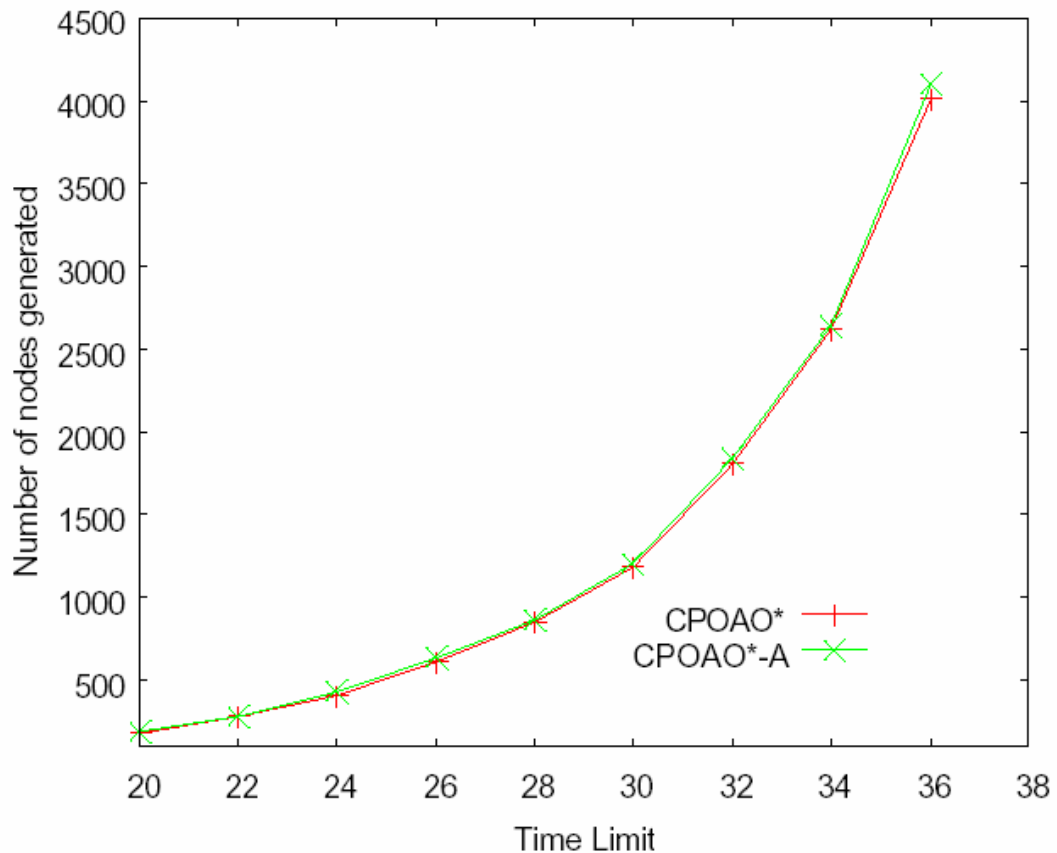
**T:** Time limit

**ER:** Expected total  
reward of the optimal  
plan

**NG:** The number of  
nodes generated

**ET:** Execution time (sec.)

# Ablation Studies - 1

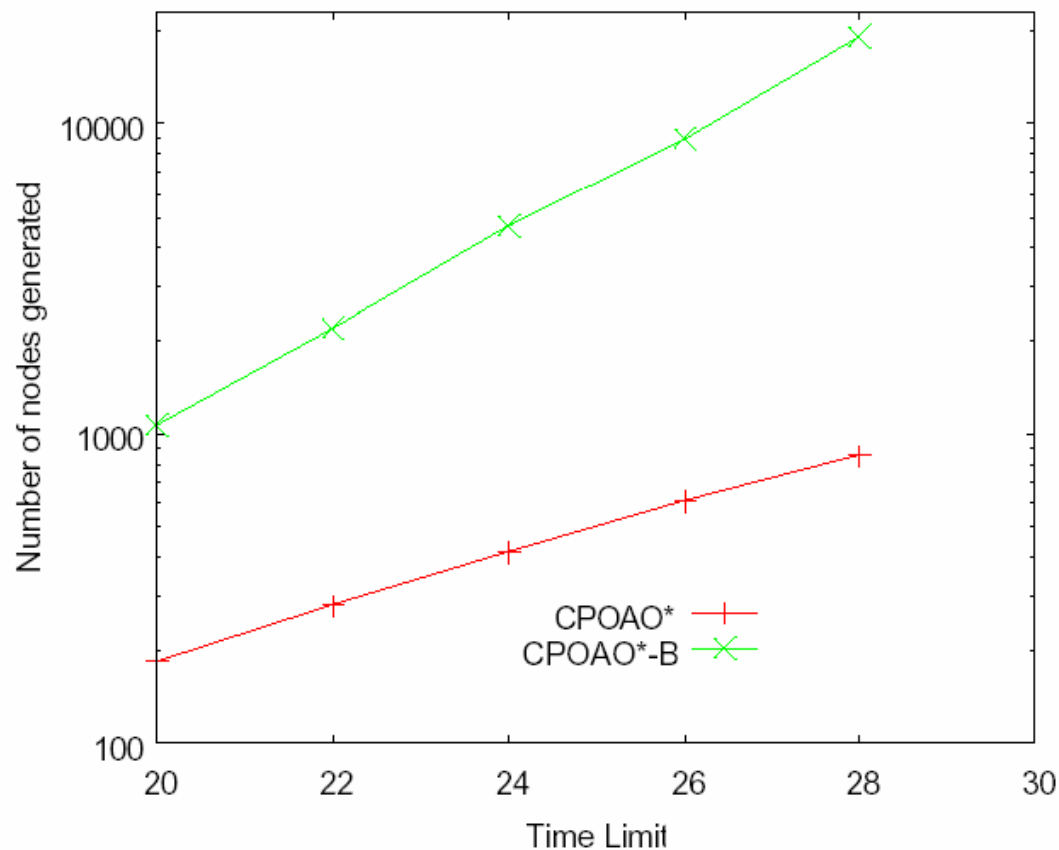


Problem of 15 locations, 16 paths and 14 rewards.

CPOAO\* uses both heuristics

CPOAO\*-A uses a constant heuristic function to estimate expected rewards for newly generated nodes.

## Ablation Studies - 2



Problem of 15 locations, 16 paths and 14 rewards.

CPOAO\*-B doesn't implement the rules of pruning the branches of concurrent actions sets (Second kind of heuristic function).



# Cache and No-Cache

Problem	Without Cache		With Cache	
	NG	ET	NG	ET
10-10-21 ( T =20 )	265	< 1	252	< 1
10-10-21 ( T =40)	27538	19	16987	7
10-17-21 ( T =20)	332	< 1	316	< 1
10-17-21 ( T =40)	102482	190	53814	58
12-12-23 ( T = 20)	301	< 1	279	< 1
12-12-23 ( T = 40)	36344	32	21911	14
12-21-23 ( T = 20)	396	< 1	348	< 1
12-21-23 ( T =40)	125959	272	72324	95
15-16-14 ( T = 20)	184	< 1	176	< 1
15-16-14 ( T = 40)	8664	3	6064	2
15-28-14 ( T = 20)	337	< 1	343	< 1
15-28-14 ( T =40)	51344	75	24982	15
5-5-10 ( T =20 )	95	< 1	93	< 1
5-5-10 ( T =40)	7404	3	4208	1
5-10-10 ( T =20)	389	< 1	320	< 1
5-10-10 ( T =40)	87854	155	59578	83

Planner with cache outperformed the planner which does not have a cache by a rate of up to 45% in terms of nodes generated.



# Conclusion

---

- An AO\* based modular framework
- Using redundant actions to increase robustness
- Ongoing work
  - Estimating state values
  - Selecting action combinations
  - Aborting actions, monitoring execution
  - Adding stochastic resource consumption
  - Improving action execution semantics



Generating Plans in Concurrent,  
Probabilistic, Oversubscribed Domains

---

**PLEASE VISIT OUR POSTER**

Li Li and Nilufer Onder  
Department of Computer Science  
Michigan Technological University