Generating Plans in Concurrent, Probabilistic, Oversubscribed Domains

Li Li and Nilufer Onder
Department of Computer Science
Michigan Technological University
(Presented by: Li Li)

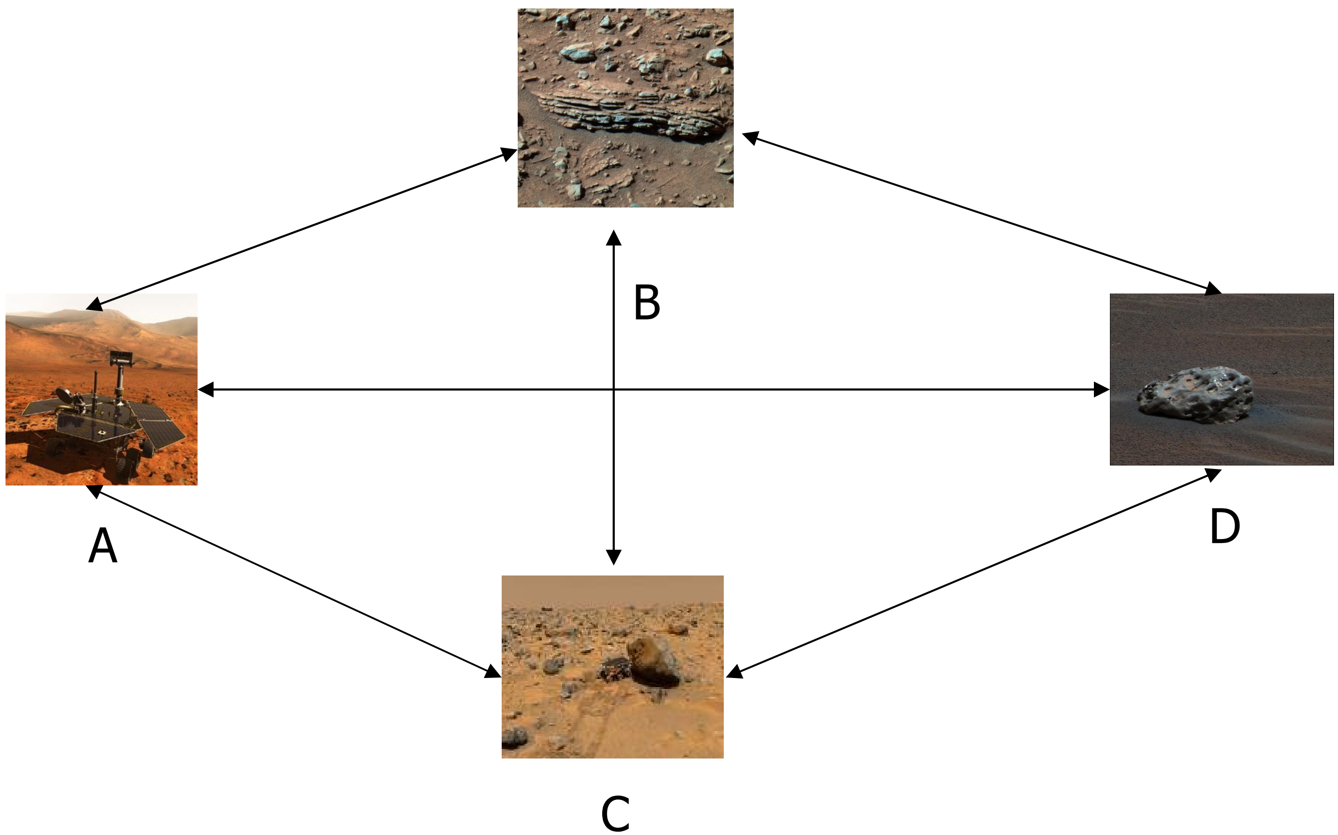
AAAI 08 Chicago
July 16, 2008

Outline

- Example domain
- Two usages of concurrent actions
- AO* and CPOAO* algorithms
- Heuristics used in CPOAO*
- Experiment results
- Conclusion and future work

A simple Mars rover domain

Locations A, B, C and D on Mars:



Main features

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

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 rewards have been achieved
- Rewards:
  - r1 = 10: Get back to location A
  - r2 = 2: Take a picture at location B
  - r3 = 1: Collect a soil sample at location B
  - r4 = 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

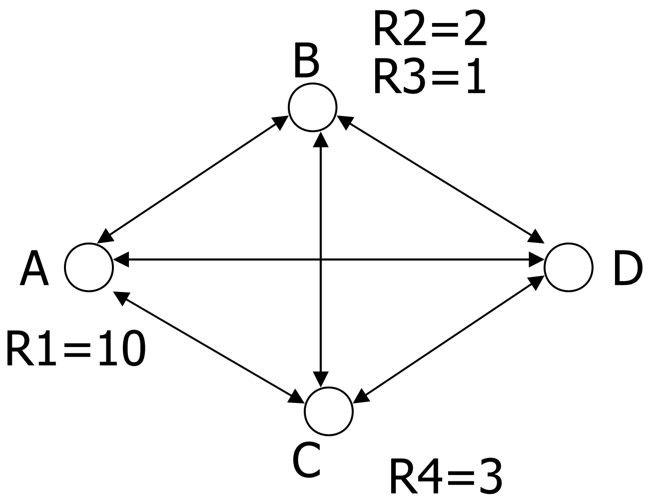
A solution to problem 1

(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

Two ways of using 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,

  execute Camera0 to achieve one reward and execute Camera1 to achieve another.
  (All finish)

  The expected total rewards
  = 10*60% + 10*50%
  = 11

Example of early finish actions

- If R1=100 and R2=10,
  Use both Camera0 and Camera1 to achieve R1. (Early finish)

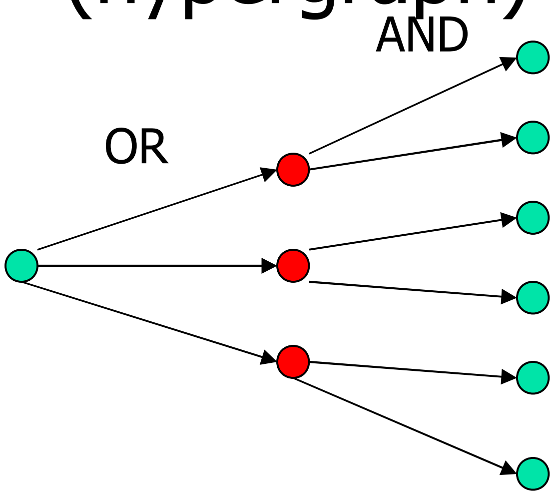
  The expected total rewards

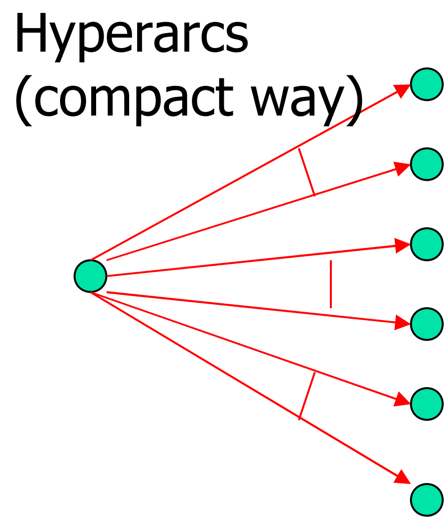
  = 100*50% + (100-100*50%)*60%

  = 50 + 30 = 80

The AO* algorithm

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





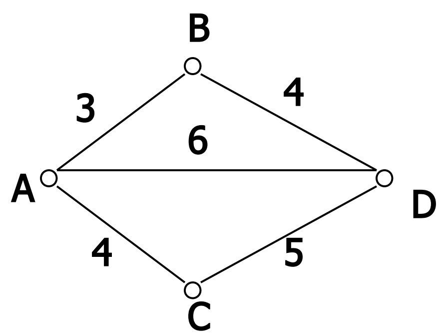
Concurrent Probabilistic Over-subscription AO* (CPOAO*)

- Concurrent action set
  - Represent parallel actions rather than individual actions
  - Use hyperarcs to represent them
- State Space
  - Resource levels are part of a state
  - Unfinished actions are part of a state

CPOAO* search Example

A Mars Rover problem

Map:



Targets:

- I1 – Image at location B
- I2 – Image at location C
- S1 – Sample at location B
- S2 – Sample at location D

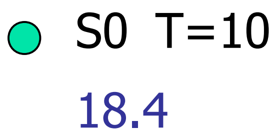
Actions:

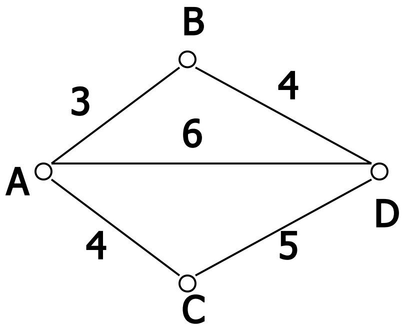
- Move (Location, Location)
- Image_S (Target) 50%, T= 4
- Image_L (Target) 60%, T= 5
- Sample (Target)   70%, T= 6

Rewards:

- Have_Picture(I1) = 3
- Have_Picture(I2) = 4
- Have_Sample(S1) = 4
- Have_Sample(S2) = 5
- At_location(A) = 10;

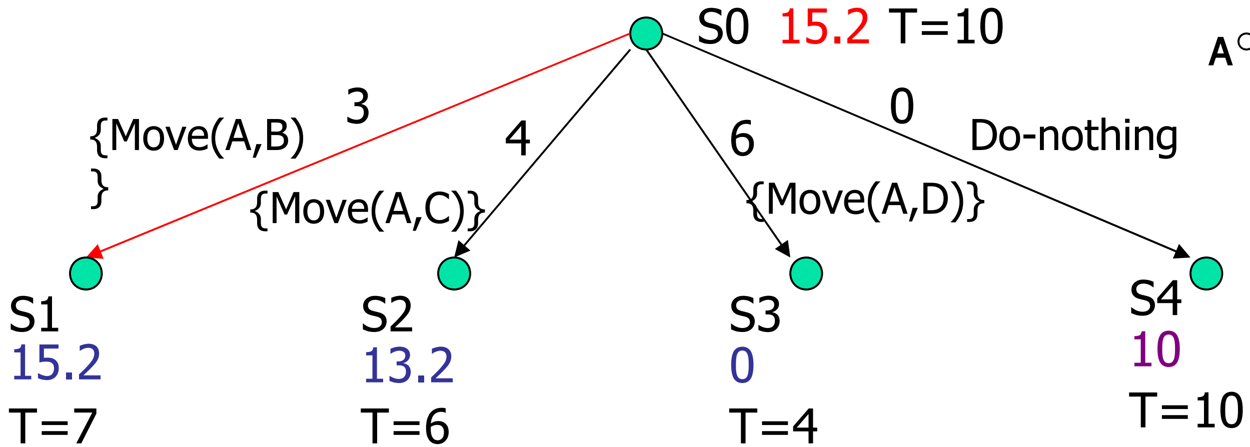
CPOAO* search Example

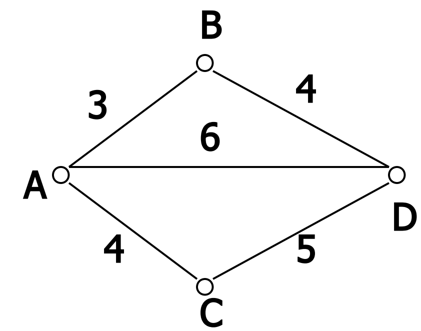


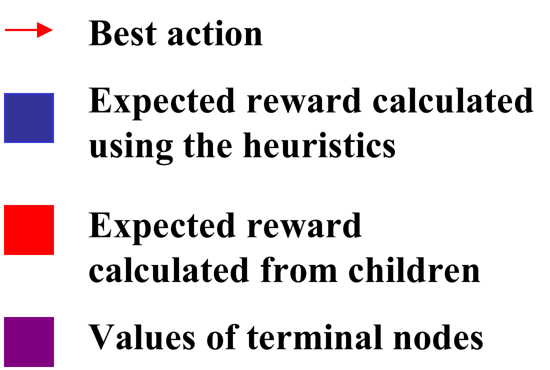


Expected reward calculated using the heuristics

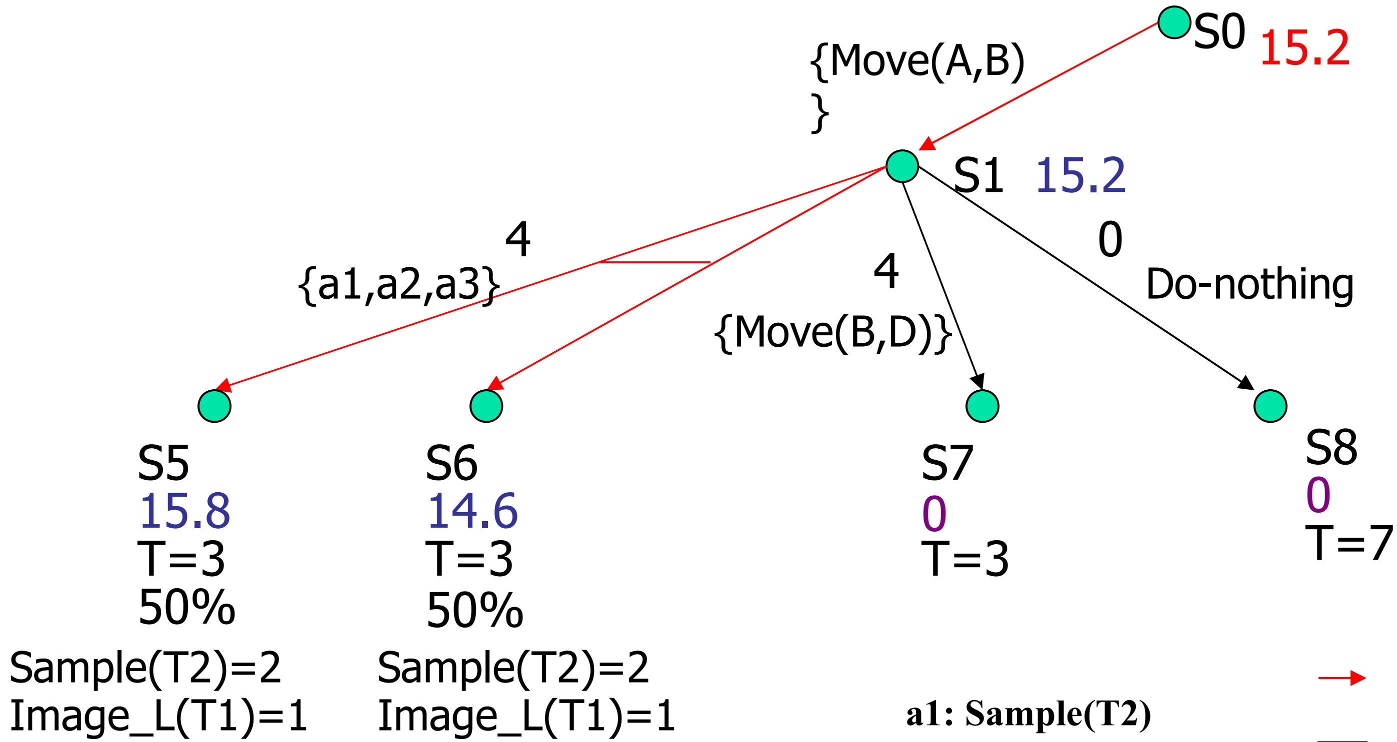
CPOAO* Search Example

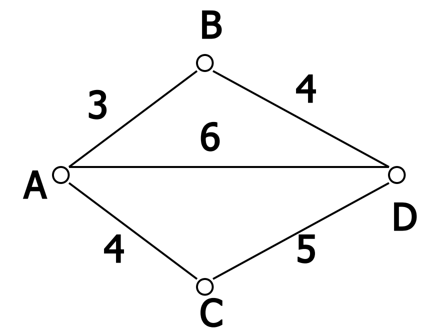






CPOAO* search Example





a1: Sample(T2)

a2: Image_S(T1)

a3: Image_L(T1)

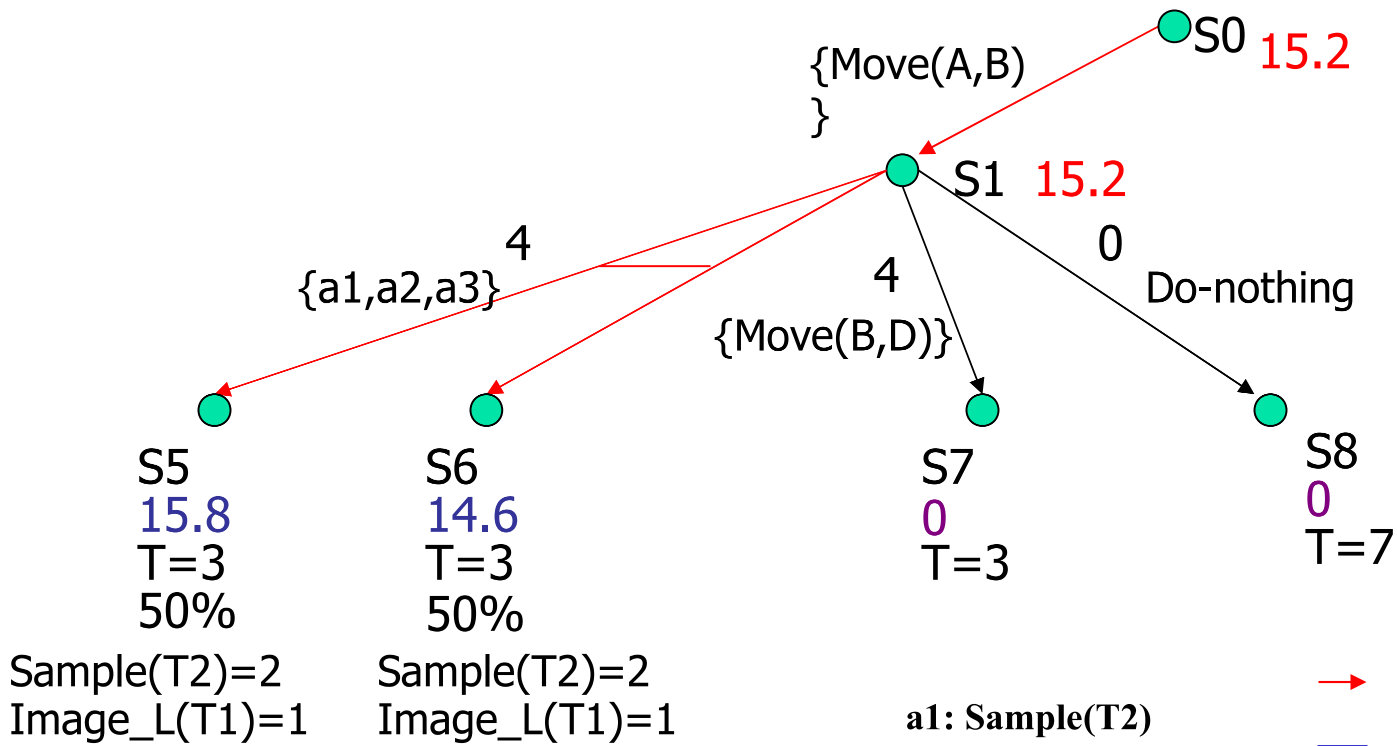
Best action

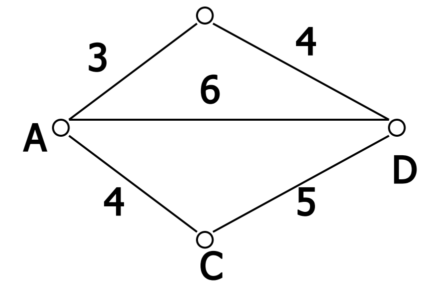
Expected reward calculated using the heuristics

Expected reward calculated from children

Values of terminal nodes

CPOAO* search Example





a1: Sample(T2)

a2: Image_S(T1)

a3: Image_L(T1)

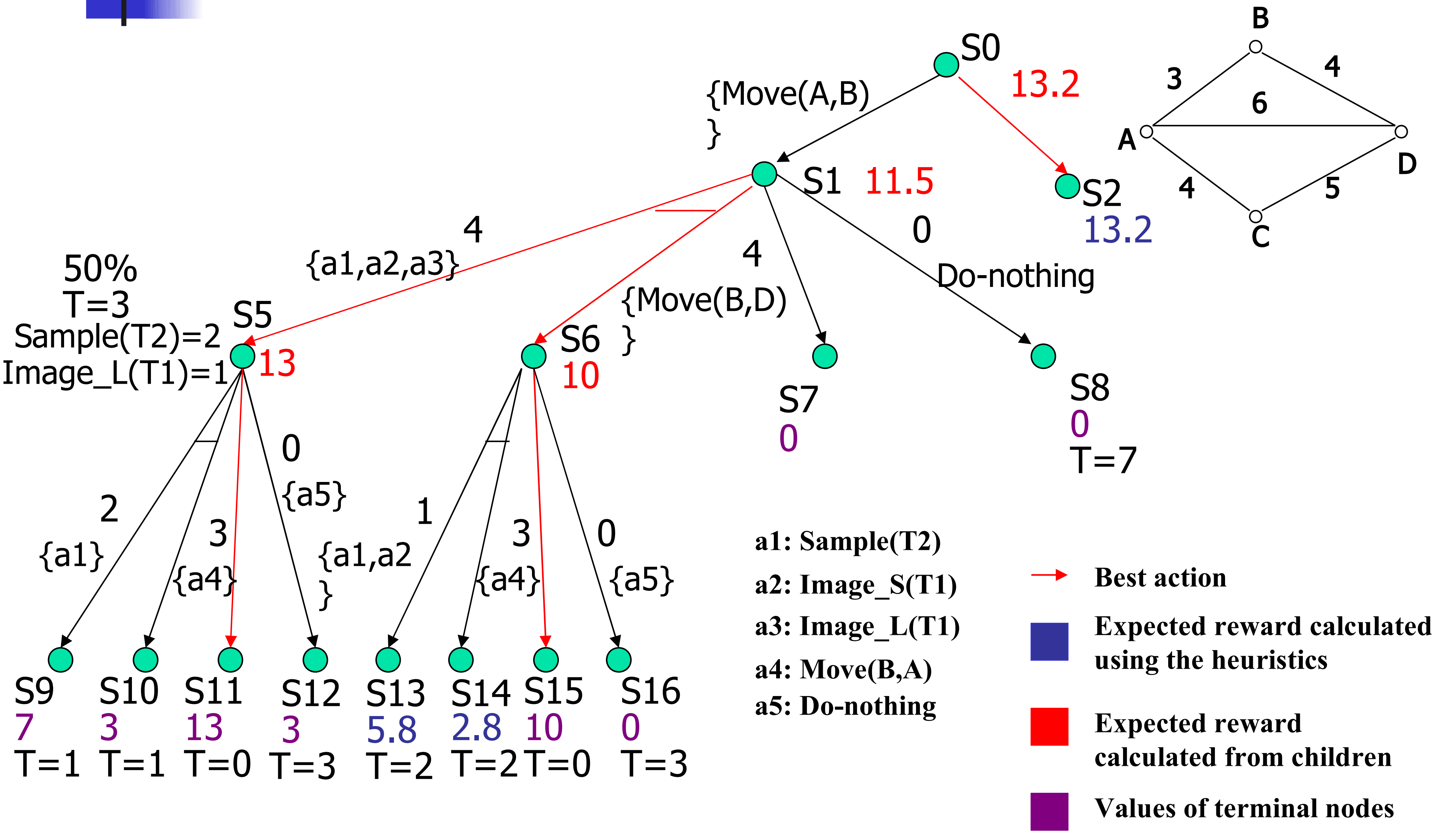
Best action

Expected reward calculated using the heuristics

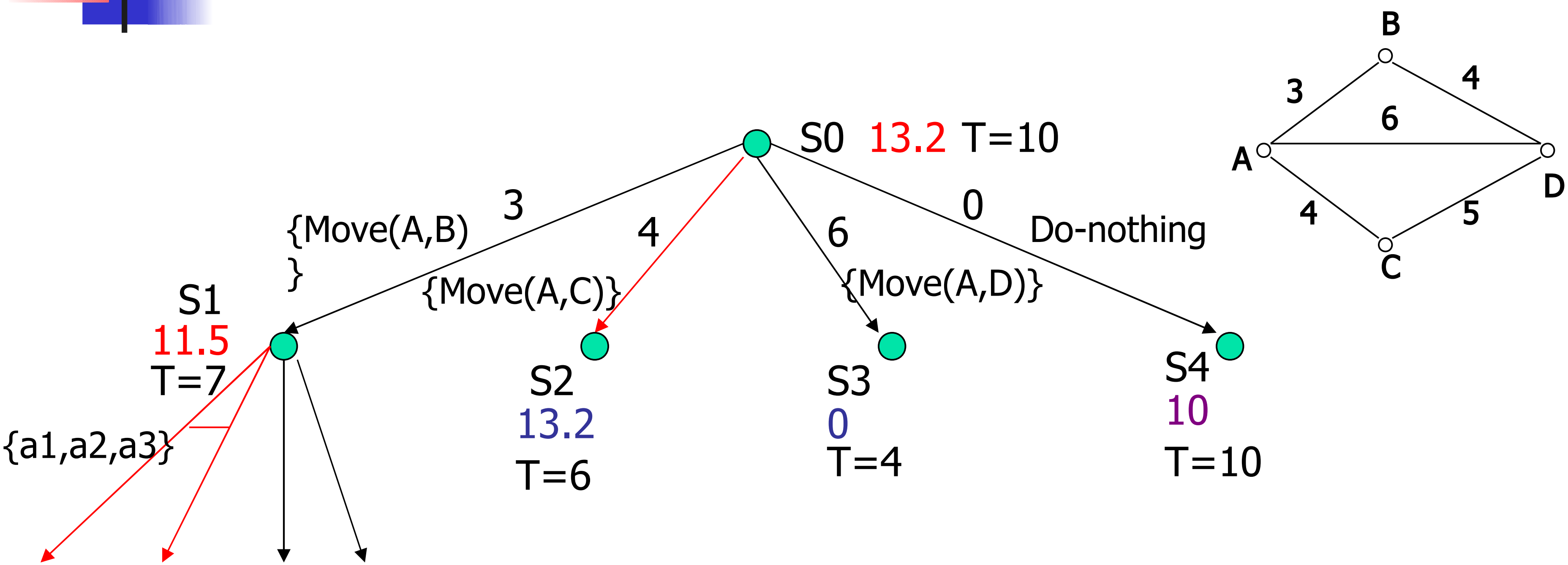
Expected reward calculated from children

Values of terminal nodes

CPOAO* search Example



CPOAO* search Example



a1: Sample(T2)

a2: Image_S(T1)

a3: Image_L(T1)

a4: Move(B,A)

a5: Do-nothing

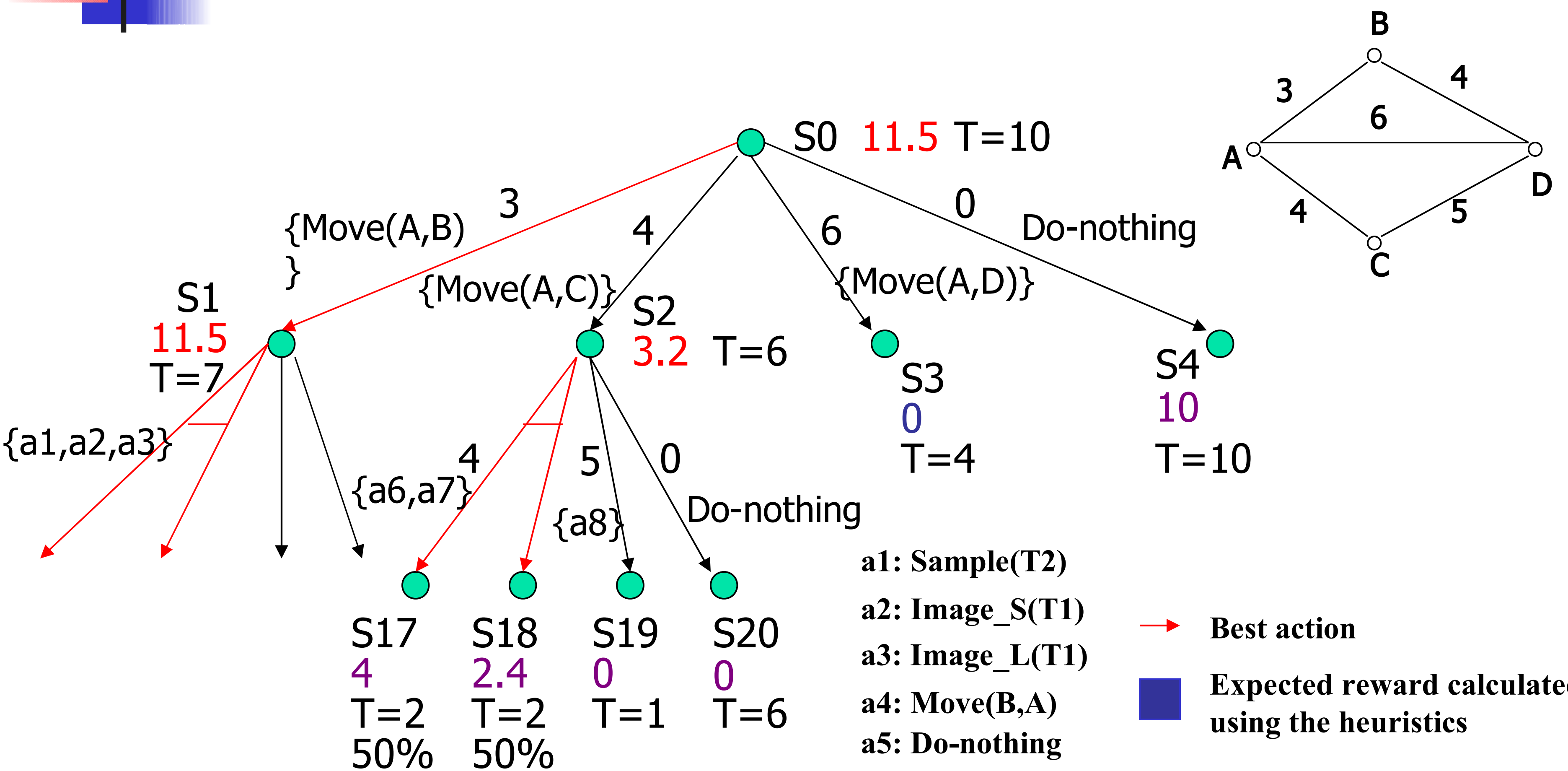
Best action

Expected reward calculated using the heuristics

Expected reward calculated from children

Values of terminal nodes

CPOAO* search Example



a1: Sample(T2)

a2: Image_S(T1)

a3: Image_L(T1)

a4: Move(B,A)

a5: Do-nothing

a6: Image_S(T3)

a7: Image_L(T3)

a8: Move(C,D)

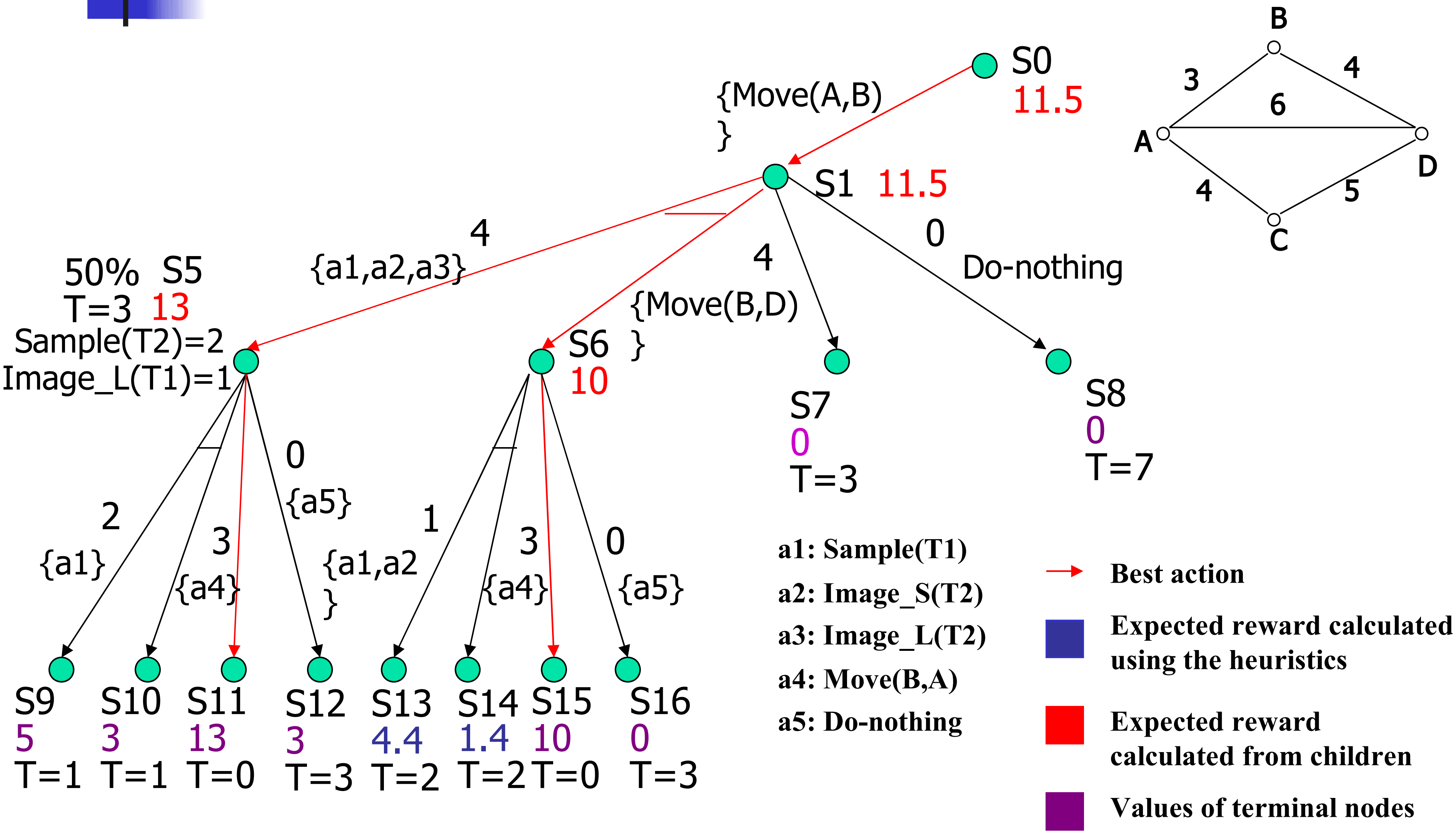
Best action

Expected reward calculated using the heuristics

Expected reward calculated from children

Values of terminal nodes

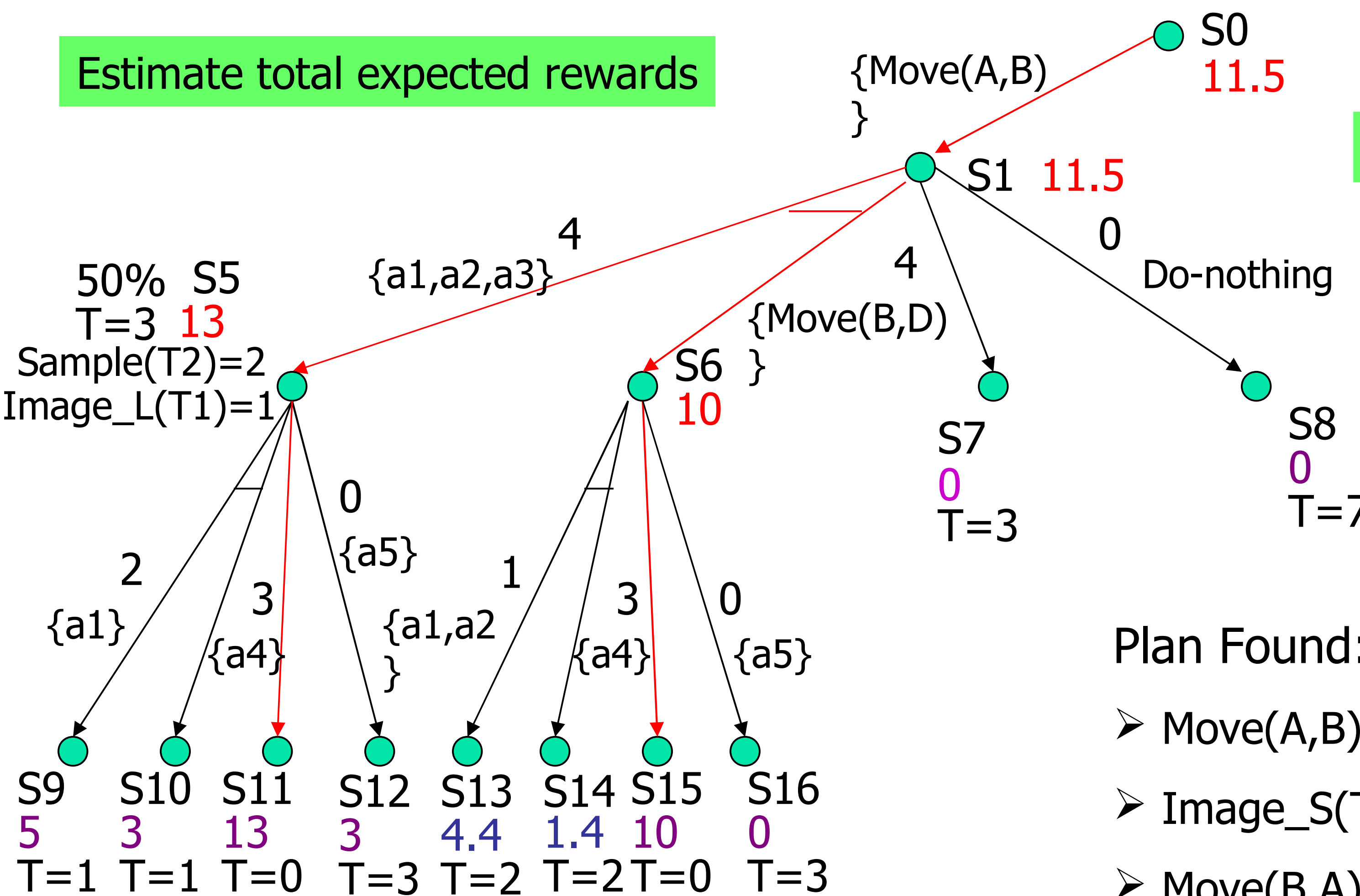
CPOAO* search Example



CPOAO* search improvements

Estimate total expected rewards

Prune branches



Plan Found:

- Move(A,B)
- Image_S(T1)
- Move(B,A)

Heuristics used in CPOAO*

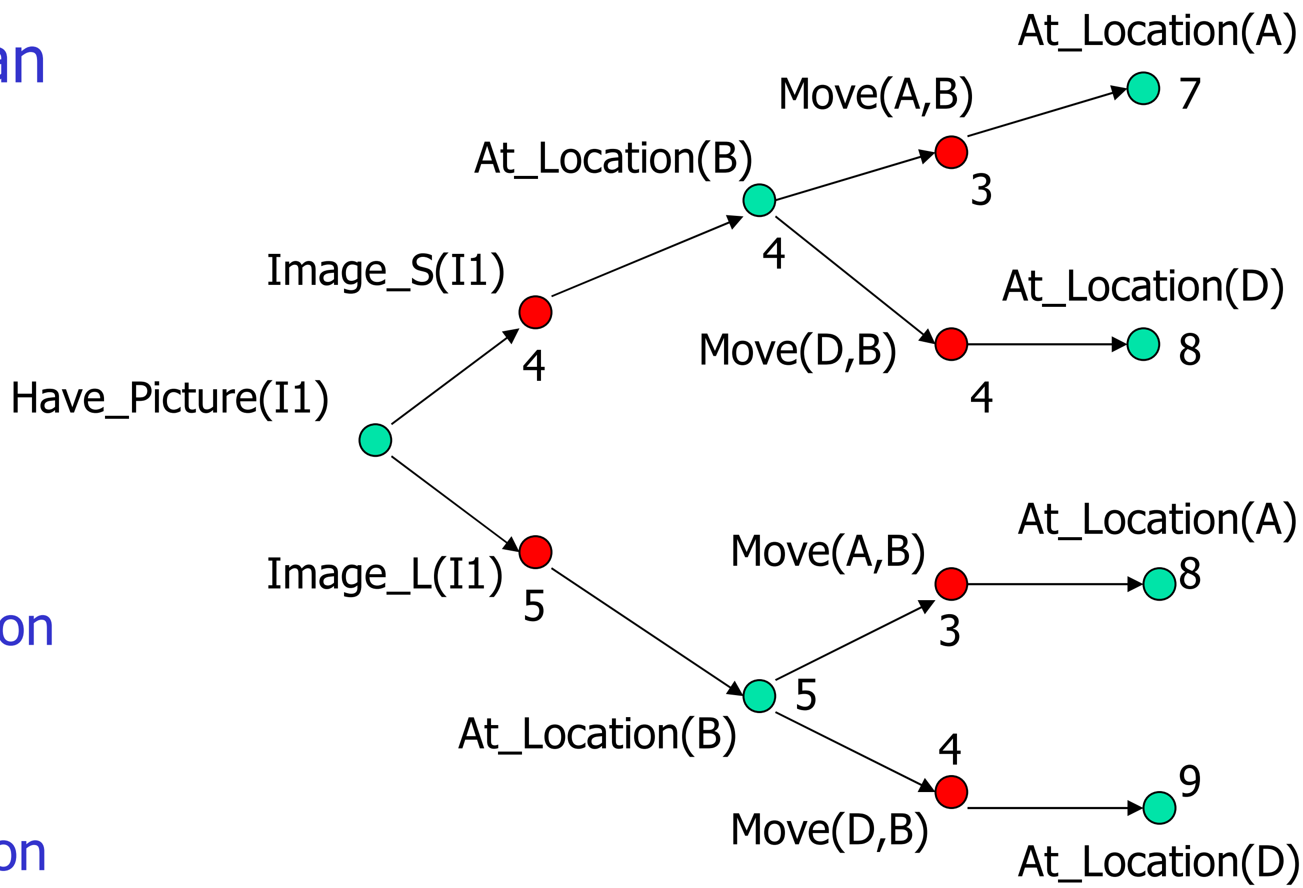
- A heuristic function to estimate the total expected reward for the newly generated states using a *reverse plan graph*.
- A group of rules to prune the branches of the concurrent action sets.

Estimating total rewards

- A three-step process using an ***rpgraph***
  1. Generate an rpgraph for each goal
  2. Identify the enabled propositions
  3. Compute the probability of achieving each goal
- Compute the expected rewards based on the probabilities
- Sum up the rewards to compute the value of this state

Heuristics to estimate the total rewards

- Reverse plan graph
- Start from goals.
- Layers of actions and propositions
- Cost marked on the actions
- Accumulated cost marked on the propositions.



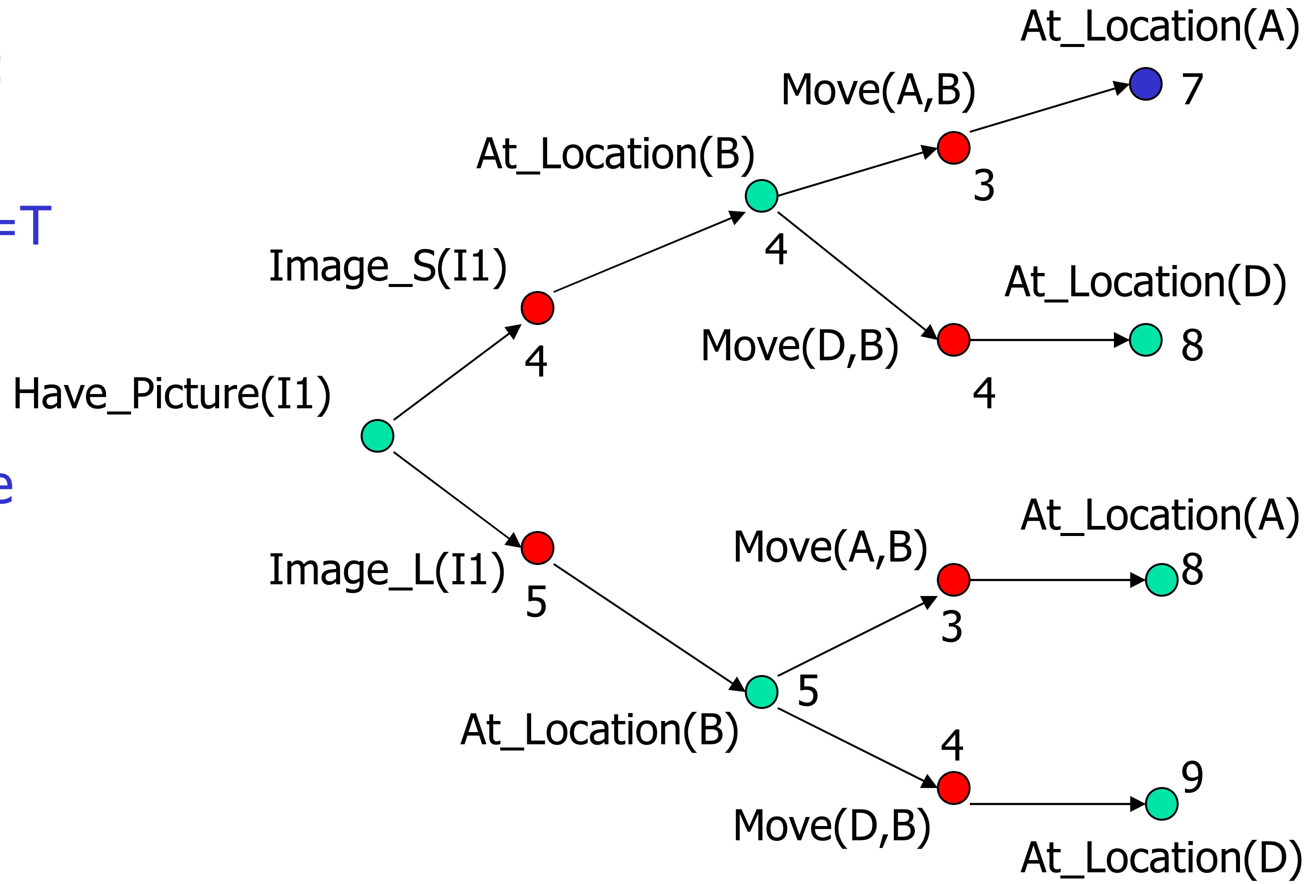
Heuristics to estimate the total rewards

- Given a specific state …

  At_Location(A)=T

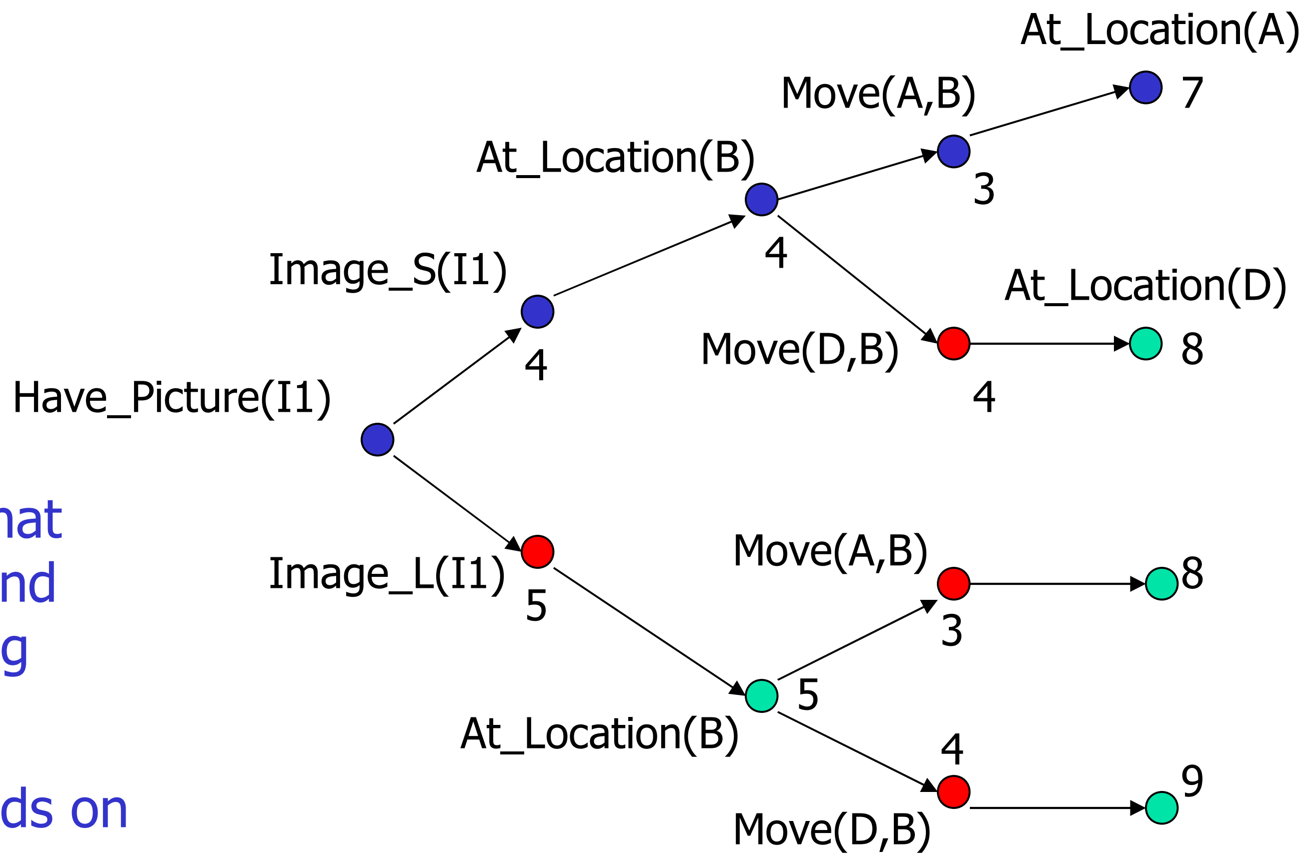
  Time= 7

- Enabled propositions are marked in blue.



Heuristics to estimate the total rewards

- Enable more actions and propositions
- Actions are probabilistic
- Estimate the probabilities that propositions and the goals being true
- Sum up rewards on all goals.



Rules to prune branches (when time is the only resource)

- Include the action if it does not delete anything

  Ex. {action-1, action-2, action-3} is better than {action-2,action-3} if action-1 does not delete anything.

- Include the action if it can be aborted later

  Ex. {action-1,action-2} is better than {action-1} if the duration of action2 is longer than the duration of action-1.

- Don't abort an action and restart it again immediately

Experimental work

- Mars rover problem with following actions
  - Move
  - Collect-Soil-Sample
  - Camera0
  - Camera1
- 3 sets of experiments - Gradually increase complexity
  - Base Algorithm
  - With rules to prune branches only
  - With both heuristics

Results of complexity experiments

Problem	Base CPOAO*		CPOAO* With Pruning Only				CPOAO* With Pruning and rpgraph			
	Time=20		Time=20		Time=40		Time=20		Time=40	
	NG	ET (s)	NG	ET (s)	NG	ET (s)	NG	ET (s)	NG	ET (s)
12-12-12	530	<0.1	120	<0.1	2832	1	56	<0.1	662	<0.1
12-12-23	1170	<0.1	287	<0.1	27914	23	86	<0.1	5269	2
12-21-12	501	<0.1	204	<0.1	11315	3	53	<0.1	1391	<0.1
12-21-23	1230	<0.1	380	<0.1	85203	99	116	<0.1	11833	5
15-16-14	1067	<0.1	180	<0.1	6306	2	60	<0.1	1232	1
15-16-31	1941	<0.1	417	<0.1	49954	61	93	<0.1	7946	2
15-28-14	1121	<0.1	347	<0.1	29760	18	71	<0.1	2460	<0.1
15-28-31	2345	<0.1	694	<0.1	---	---	146	<0.1	19815	8

Problem: locations-paths-rewards; **NG**: The number of nodes generated; **ET**: Execution time (sec.)

Ongoing and Future Work

- Ongoing
  - Add typed objects and lifted actions
  - Add linear resource consumptions
- Future
  - Explore the possibility of using state caching
  - Classify domains and develop domain-specific heuristic functions
  - Approximation techniques

Related Work

- LAO*: A Heuristic Search Algorithm that finds solutions with loops (Hansen & Zilberstein 2001)
- CoMDP: Concurrent MDP (Mausam & Weld 2004)
- GSMDP: Generalized semi-Markov decision process (Younes & Simmons 2004)
- mGPT: A Probabilistic Planner based on Heuristic Search (Bonet & Geffner 2005)
- Over-subscription Planning (Smith 2004; Benton, Do, & Kambhampati 2005)
- HAO*: Planning with Continuous Resources in Stochastic Domains (Mausam, Benazera, Brafman, Meuleau & Hansen 2005)

Related Work

- CPTP:Concurrent Probabilistic Temporal Planning (Mausam & Weld 2005)
- Paragraph/Prottle: Concurrent Probabilistic Planning in the Graphplan Framework (Little & Thiebaux 2006)
- FPG: Factored Policy Gradient Planner (Buffet & Aberdeen 2006)
- Probabilistic Temporal Planning with Uncertain Durations (Mausam & Weld 2006)
- HYBPLAN:A Hybridized Planner for Stochastic Domains (Mausam, Bertoli and Weld 2007)

Conclusion

- An AO* based modular framework
- Use redundant actions to increase robustness
- Abort running actions when needed
- Heuristic function using reverse plan graph
- Rules to prune branches