Simple and Fast Strong Cyclic Planning for Fully-Observable Nondeterministic Planning Problems

UCO UNIVERSITY OF CENTRAL OKLAHOMA

UTD The University of Texas at Dallas

Jicheng Fu[1], Vincent Ng[2], Farokh Bastani[2], and I-Ling Yen[2]

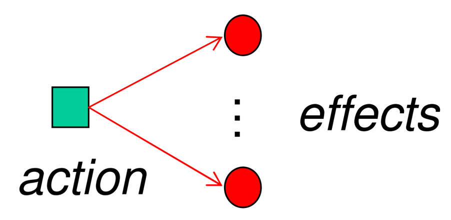
[1]The University of Central Oklahoma [2]The University of Texas at Dallas

Problem

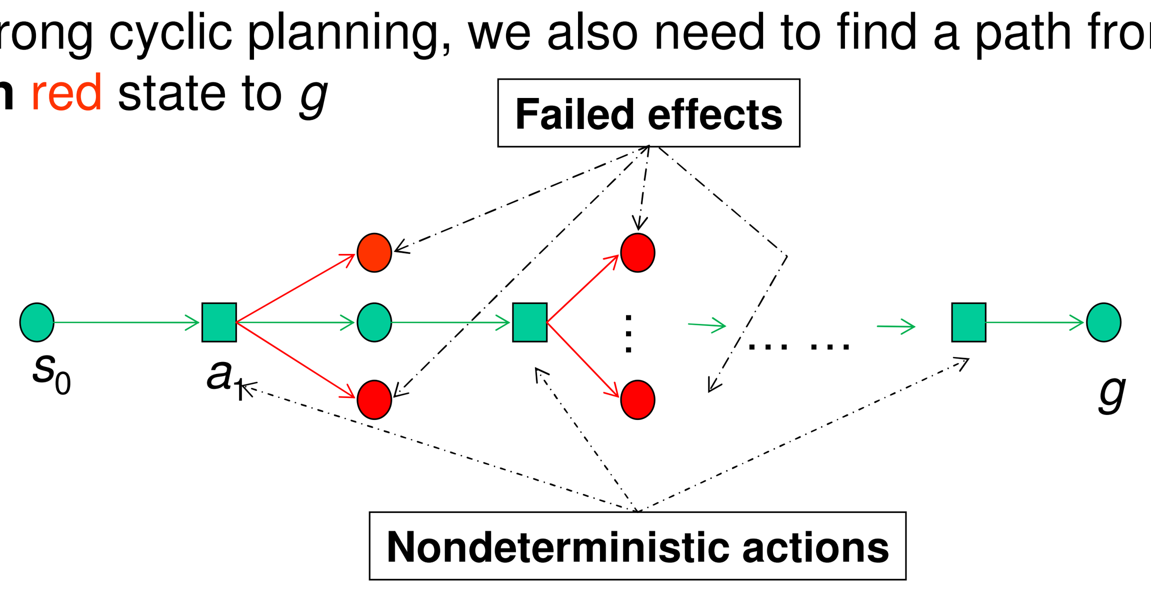
- Find strong cyclic solutions to Fully-Observable Nondeterministic (FOND) planning problems

Related Concepts

- In **nondeterministic** planning
  - an action may generate multiple effects



- In **fully-observable** planning
  - the states of the world are fully observable
- More challenging than finding **weak** plans
  - **Weak plans**: only need to establish one path from the initial state to the goal state
  - **Strong cyclic plans**: need to establish one path from each state reachable from the initial state to the goal state
- **Example**: Given initial state s_0 and goal g,
  - the green path is a weak plan, since it is **one** path from s_0 to g
  - in strong cyclic planning, we also need to find a path from **each** red state to g

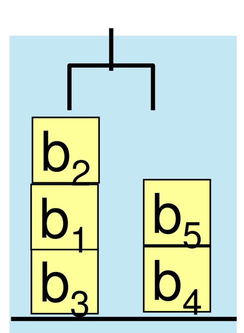


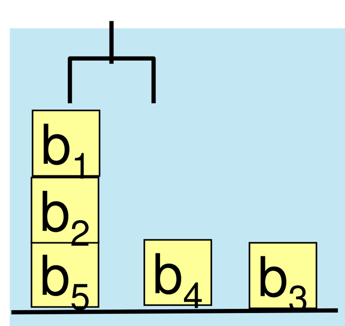
  - An outcome of an action that is included in the weak plan (i.e., a green state) is its **intended effect**
  - An outcome of an action that is not included in the weak plan (i.e., a red state) is a **failed effect** of the action

Basic Strong Cyclic Algorithm

- **3 steps**
  1. Generate a weak plan from s_0 to g.
  2. For each failed effect e, recursively find a weak plan from e to g.
  3. If a dead end is met (i.e., no path leads to g from it), then *backtrack* (i.e., disable the action that leads to the dead end and try another path)
- **Example: Blocksworld**

Initial state (s_0)





Goal state (g)

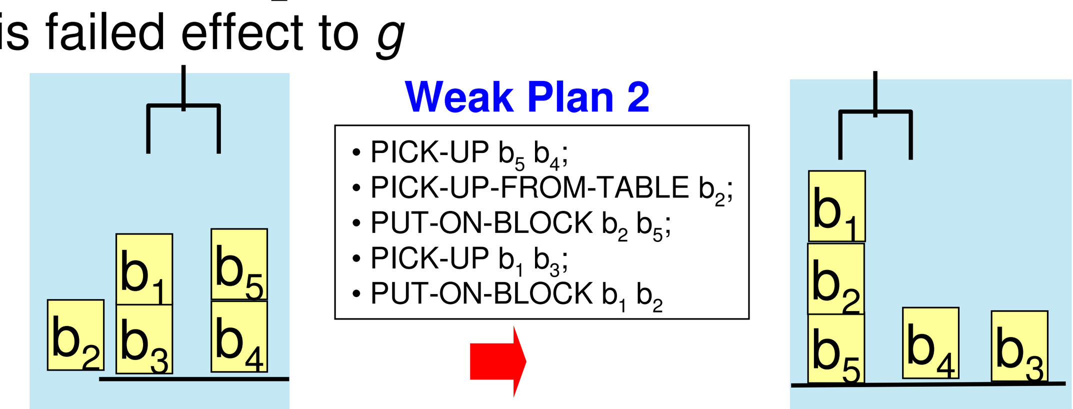
  - To generate a strong cyclic plan:

Step 1: Find a weak plan from s_0 to g

Weak Plan 1

- PICK-UP b_2 b_1;
- PUT-ON-BLOCK b_2 b_5;
- PICK-TOWER b_2 b_5 b_4;
- PUT-TOWER-DOWN b_2 b_5;
- PICK-UP b_1 b_3;
- PUT-ON-BLOCK b_1 b_2

Step 2: Since action PICK-UP b_2 b_1 may generate the failed effect of dropping b_2 onto the table, we generate a weak plan from this failed effect to g



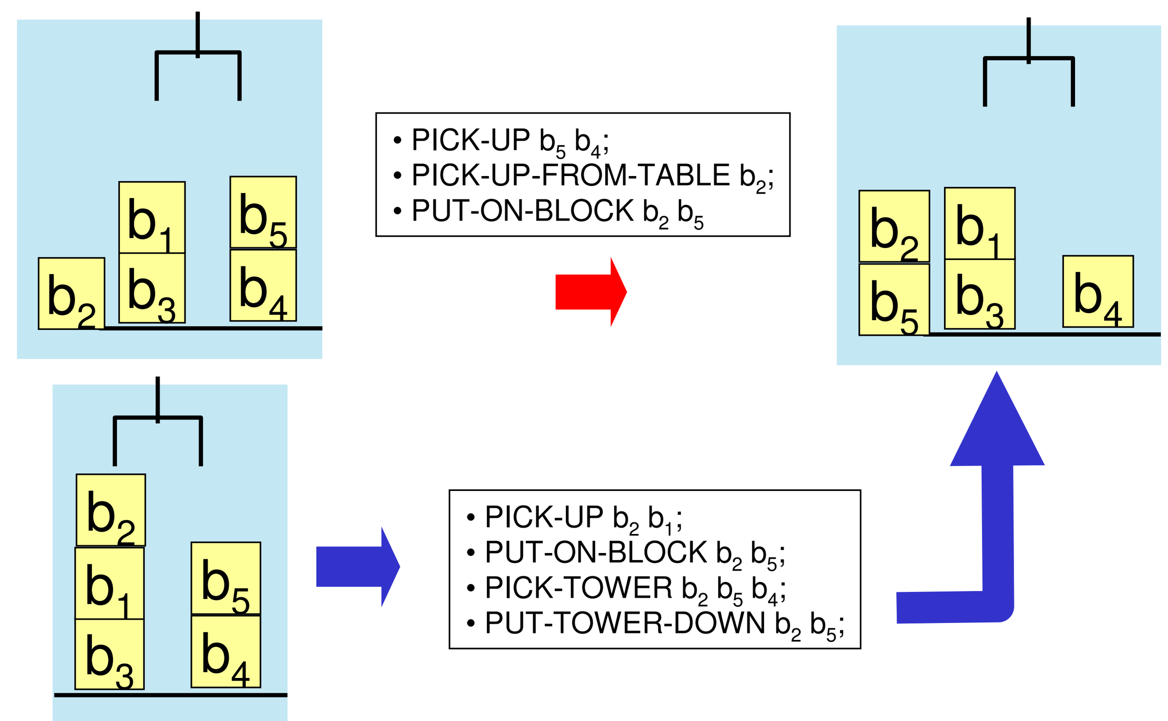
Step 3: Since no dead-ends are found, no backtrack is needed

- This Basic algorithm is **inefficient**
  - Certain states are repeatedly explored: the last two actions of Weak Plan 1 and Weak Plan 2 are identical.
- **Goal**: Improve the efficiency of the Basic algorithm by proposing two extensions

Two Extensions

Extension 1: State Reuse

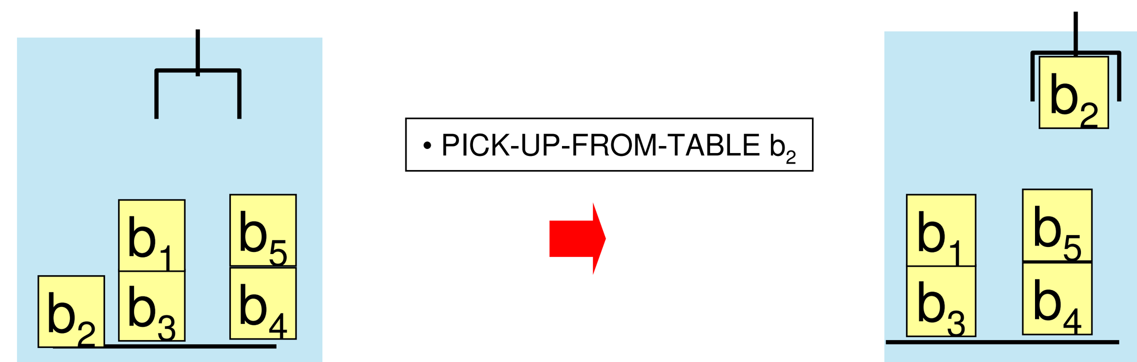
- **Observation 1**: the first three actions in Weak Plan 2 yield the same state (call it s) as the first four actions in Weak Plan 1



- **Observation 2**: since s is already solved in Weak Plan 1, there is no need to try to find a path from s to g in Weak Plan 2
- Given these observations, state reuse aims to improve efficiency by **stopping the search** as soon as a solved state is reached

Extension 2: Goal Alternative

- **Observation**: To handle a failed effect (e.g., b_2 falling onto the table for action PICK-UP b_2 b_1), instead of establishing a path to the ultimate goal g (as in the Basic algorithm), we can try to establish a path to **intended effect** of PICK-UP b_2 b_1, i.e., **holding b_2**



  - The plan contains a **single** action, PICK-UP-FROM-TABLE b_2
    - **Planning efficiency is improved and plan size is reduced!**
  - However, if a path to the intended effect cannot be found, we can then try to establish a path to the original goal g.
- This is the **goal alternative** heuristic: it aims to improve planning efficiency and reduce plan size by searching for an **alternative**, **presumably closer goal**, the intended effect of an action, and backing off to the original goal if needed

Evaluation

- **Goal**: Evaluate FIP, which implements the Basic algorithm together with our two extensions, on problem instances from 4 domains in the IPC2008 FOND track
  - Blocksworld, faults, first-responders, forest
  - Compared against two state-of-the-art planners: Gamer & MBP
- **Results and Discussion**
  - FIP has a better problem coverage than Gamer & MBP [Table 1]
    - Gamer & MBP cannot solve more than 10 problems in Blocksworld
    - FIP can solve all problems efficiently (cutoff time 1,200 seconds)
  - FIP outperforms other planners w.r.t. CPU time t (expressed in seconds) and solution size s (expressed in the number of states in the solution policy) [Table 2]

Domain	Gamer	MBP	Basic	FIP
blocksworld (30)	10	1	30	30
faults (55)	38	16	55	55
first-responders (100)	21	11	75	75
forest(90)	7	0	7	7
Total (275)	76	28	167	167

Table 1

Problem	Gamer		MBP	Basic		FIP	
	t	*s*	*t*	*t*	*s*	*t*	*s*
bw-1	38.748	10	*3556.517*	0.011	12	0.007	8
bw-5	37.506	13	---	0.020	12	0.010	12
bw-10	28.650	13	---	0.015	22	0.009	10
bw-12	---	---	---	3.507	225	0.285	45
bw-25	---	---	---	452.642	537	59.230	312
bw-30	---	---	---	42.755	102	3.126	48
faults-7-7	90.996	235	---	0.043	258	0.005	23
faults-8-8	1106.105	325	---	0.101	514	0.007	26
faults-9-9	830.272	511	---	0.217	848	0.007	29
faults-10-10	---	---	---	0.859	2050	0.009	32
f-r-2-3	0.142	12	63.388	0.003	11	0.003	11
f-r-4-2	0.118	8	---	0.003	6	0.003	6
f-r-6-2	1.016	7	---	0.003	7	0.003	7
forest-2-6	4.769	50	---	0.008	50	0.008	50
forest-2-7	8.122	44	---	0.007	44	0.008	44
forest-2-8	0.638	56	---	0.007	56	0.008	56
forest-2-9	0.607	42	---	0.006	42	0.007	42
forest-2-10	0.927	44	---	0.007	44	0.008	44

Table 2