SAT based Classical Planning for Multi-Agent Systems

Ravi Kuril

thesis guide
Dr. Indranil Saha

Department of Computer Science
IIT Kanpur

July 10, 2018

Outline

Outline

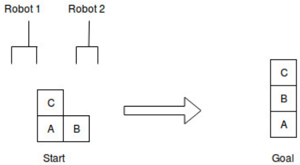
Definitions

- **AI Planning:**
  The realization of strategies or action sequences by intelligent agents to achieve a goal is called planning.
- **Classical planning:**
  The environments that are fully observable, deterministic, finite, static, and discrete (in time, action, objects, and effects).
- **Multiagent Classical planning:**
  Multiple agents performs a sequence of actions to obtain a set of goals.
- **Make-span:**
  The makespan is the total horizon length of the plan.
- **Total Cost:**
  Summation of the cost of each actions in the plan.
- **Time step N:**
  In the N Number of iterations, an agent can execute a plan of maximum trajectory of N length.

Objective of the Thesis

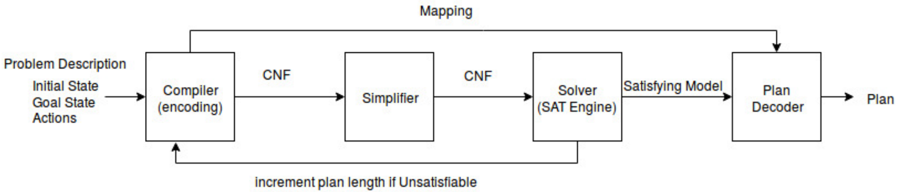
- Existing tools are fast but compromise in optimal make-span.
- Introduce a SAT based encoding approach for multi-agent classical planning problem
- Develop a SAT based tool which can solve multi-agent classical planning problems.
- Optimal make-span.

Multi-agent Classical Planning, Existing Tools



- Multi-agent Classical Planning example : blockworld
- Graph based planner :
    - ADP and ADP-Legacy, MAPlan
- Reduction based planner :
    - Madagascar, BlackBox

SAT Planning Architecture



Outline

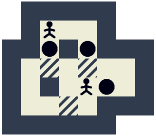


Figure: Ma-Sokoban puzzle

- MA-Sokoban: Multiple agents push boxes to place them on goal locations.
- Encoding Approach 1

1. Player movement constraints:
$p_i(x, y, t) \Rightarrow p_i(x, y, t+1) \vee p_i(x+1, y, t+1) \vee p_i(x-1, y, t+1) \vee p_i(x, y+1, t+1) \vee p_i(x, y-1, t+1)$
where $\forall p_i, p_i \in R$

2. Box push movement constraints: Right Push

$$b_i(x, y, t+1) \Rightarrow (b_i(x, y, t) \vee (\bigvee_{r \in R} (p_r(x-2, y, t) \wedge b_i(x-1, y, t) \wedge p_r(x-1, y, t+1))))$$

3. Player Head-on constraint:

$$\neg((p_1(x, y, t) \wedge p_2(x, y+1, t) \wedge p_1(x, y+1, t+1) \wedge p_2(x, y, t+1))$$

$$\neg((p_1(x, y, t) \wedge p_2(x+1, y, t) \wedge p_1(x+1, y, t+1) \wedge p_2(x, y, t+1))$$

4. Player's single place existence constraints:
$\neg(p_i(x_1, y_1, t) \wedge p_i(x_2, y_2, t))$ where $\forall p_i, p_i \in R$ and $t \in T$.
5. Box's single place existence constraints:
$\neg(b_i(x_1, y_1, t) \wedge b_i(x_2, y_2, t))$ where $\forall b_i, b_i \in S$ and $t \in T$.

- Constraints:
  6. Player to Player collision constraints:

$$\neg(p_i(x, y, t) \wedge p_j(x, y, t))$$

where $\forall p_i, p_i \in R$,$\forall p_j, p_j \in R$, $i \neq j$ and $t \in T$.
7. Box to Box collision constraints:

$$\neg(b_i(x, y, t) \wedge b_j(x, y, t))$$

where $\forall b_i, b_i \in S$,$\forall b_j, b_j \in S$,$i \neq j$ and $t \in T$.
8. Box to player collision constraints

$$\neg(b_i(x, y, t) \wedge p_j(x, y, t))$$

where $\forall b_i, b_i \in S$,$\forall p_j, p_j \in R$ and $t \in T$.
9. Obstacle collision constraints:
Player with obstacle:

$$\neg p(x_o, y_o, t)$$

where (x_o, y_o) is the obstacle location.
Box with obstacle:

$$\neg b(x_o, y_o, t)$$

where (x_o, y_o) is the obstacle location.

- **Constraints:**
  1. Box push movement constraints
  Push Right:

$$(b_i(x, y, t) \wedge b_i(x+1, y, t+1) \Rightarrow (\bigvee_{(p_r \in R)} (p_r(x-1, y, t) \wedge p_r(x, y, t+1))$$

  where $\forall b_i, b_i \in S$. Similarly all the constraints are generated for left, up and down direction.
  2. Box movement constraints:
  $b_i(x, y, t) \Rightarrow b_i(x, y, t+1) \vee b_i(x+1, y, t+1) \vee b_i(x-1, y, t+1) \vee b_i(x, y+1, t+1) \vee b_i(x, y-1, t+1)$
  where $\forall b_i, b_i \in S$

Motivating Example Cont. Encoding Approach 1 vs Encoding Approach 2

- **Constraints: Push Right constraint for one box and two players**
  Encoding Approach 1

$$b_1(x, y, t+1) \Rightarrow (b_1(x, y, t) \vee (p_1(x-2, y, t) \wedge b_1(x-1, y, t) \wedge p_1(x-1, y, t+1)) \bigvee$$

$$(p_2(x-2, y, t) \wedge b_1(x-1, y, t) \wedge p_2(x-1, y, t+1)))$$

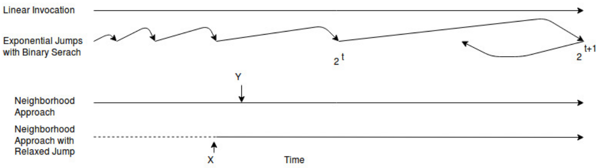
  Boolean Simplification: **9** clauses
  Encoding Approach 2
  Proposition 1

$$(b_1(x, y, t) \wedge b_1(x+1, y, t+1) \Rightarrow ((p_1(x-1, y, t) \wedge p_1(x, y, t+1)) \bigvee (p_2(x-1, y, t) \wedge p_2(x, y, t+1)))$$

  Proposition 2

$$b_1(x, y, t) \Rightarrow b_1(x, y, t+1) \vee b_1(x+1, y, t+1) \vee b_1(x-1, y, t+1) \vee b_1(x, y+1, t+1) \vee b_1(x, y-1, t+1)$$

  Boolean Simplification: **5** clauses (4 clauses in CNF of proposition 1, 1 clause of proposition 2)



- Solver Invocation Strategies
  Strategy 1: Linear Approach (Encoding approach 1+PBL Interface)
  Strategy 2: Exponential Jumps Approach (Encoding Approach 2)
- Clause Optimization Strategy
  Strategy 1: Neighborhood Encoding Approach (Encoding Approach 2)
  Strategy 2: Neighborhood Encoding Approach with Relaxed Jump(Encoding Approach 2)

Outline

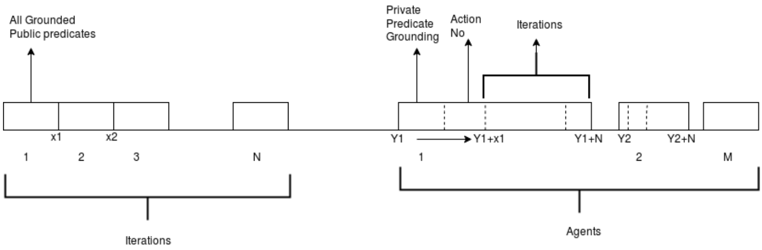
MA-PDDL representation of Blockworld domain

```
(define(domain blocks)
    (:requirements :typing :multi-agent :unfactored-privacy)
(:types
    agent block - object
)
(:predicates
    (on ?x - block ?y - block)
    (ontable ?x - block)
    (clear ?x - block)
    (:private ?agent - agent
        (holding ?agent - agent ?x - block)
        (handempty ?agent - agent)
    )
)
(:action pick-up
    :agent ?a - agent
    :parameters (?x - block)
    :precondition (and(clear ?x)(ontable ?x)(handempty ?a))
    :effect (and (not (ontable ?x)) (not (clear ?x)) (not (handempt
            (holding ?a ?x) )
)
(:action put-down
....
....
)
(:action stack
)
(:action unstack
....
....
)
```

Problem File

```
(define (problem BLOCKS-4-0) (:domain blocks)
    (:objects
        a - block
        c - block
        b - block
    (:private a1
        a1 - agent
    )
    (:private a2
        a2 - agent
    )
)
(:init
    (handempty a1)
    (handempty a2)
    (clear c)
    (ontable c)
    (ontable b)
    (on a b)
)
(:goal
    (and
        (on b a)
        (on c b)
    )
```

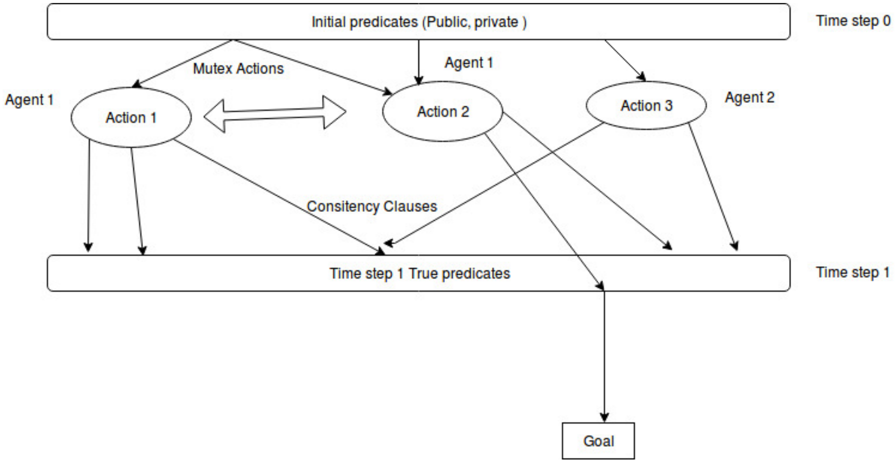
Automated Reduction Model



- Grounding: Generate all cases of a predicate with objects.
- Mutex generation: Conflict actions.(two agents pick one box)
- Consistency Constraints: More than one action's effect is same.(cover a tile)

Automated Reduction Model cont..

- **Steps of the Algorithm**
  Step 1: Ground all public predicates and assign them a literal number.
  Step 2: Assign each agent a starting number. Initialize current time step to one.
  Step 3: Generate all possible moves of one-time step for all the agents and store those clauses.
  Step 4: Generate mutex action clauses for all the actions invoked by an agent.
  Step 5: Create clauses of the initial conditions and store them.
  Step 6: If the current time step is one then go to step 7 otherwise remove goal clauses of previous time step which is stored.
  Step 7: Create goal clauses for the current time step and store them.
  Step 8: If the current time step is one then go to step 9 otherwise shift actions clauses and mutex clauses for one more time step for all the agents and store them.
  Step 9: Call SAT solver and provide all the clauses which are generated in above steps.
  Step 10: If the SAT solver gives satisfiable as a result. Extract the plan and return. otherwise, go to step 6.


Initial predicates (Public, private)
Time step 0
Mutex Actions
Agent 1
Agent 1
Action 1
Action 2
Action 3
Agent 2
Consitency Clauses
Time step 1 True predicates
Time step 1
Goal

Automated Reduction Model cont.. Constraints generation

- **Initial Configuration clauses**
- **Action Clauses:** for every action $a =< pre(a), add(a), del(a) >$

$$\bigvee_{i}^{add(a)} (add(a)pred^{i}_{(t+1)} \rightarrow add(a)pred^{add(a)\setminus i}_{(t+1)} \wedge del(a)_{pred(t+1)} \wedge$$

$$pre(a)pred_{(t)} \wedge Act_{no}(a)_{(t)} \wedge Ins_{no}(a)_{(t)})$$

similarly if the negative goal predicates are allowed.

$$\bigvee_{i}^{del(a)} (del(a)pred^{i}_{(t+1)} \rightarrow del(a)pred^{add(a)\setminus i}_{(t+1)} \wedge add(a)_{pred(t+1)} \wedge$$

$$pre(a)pred_{(t)} \wedge Act_{no}(a)_{(t)} \wedge Ins_{no}(a)_{(t)})$$

- **Constraints:**
  Predicate Clauses:
  $[Eff\_pred_i(a_i, j, k, t+1) \rightarrow Eff\_pred_i(a_i, j, k, t)$

$$\bigvee_{(\theta=1)}^{K} \Big(\bigvee_{(j=1, Eff\_pred_i(j,k,t+1) \in A_\theta)}^{A_\theta} \Big(\Big(\bigwedge_{\phi=1, \phi \neq i}^{|a_{(j,\theta)}(eff\_pred)|} Eff\_pred_\phi(a_{(j,\theta)}, p, q, t+1) \wedge$$

$$\Big(\bigwedge_{\psi=1}^{|a_{(j,\theta)}(pre\_pred)|} Pre\_pred_\psi(a_{(j,\theta)}, r, s, t)\Big)\Big) \wedge Inst_{no} \wedge Act_{(no, A_\theta)}\Big)\Big)]$$

where a_i denotes ith action of agent i. p, r is predicate of an action. q is the qth instance of the pth predicate. t is the time step. A_θ is the action applied by the θ agent. $|a_i(eff\_pred)|$ is the number of effect predicates of an action of ith agent. $|a_i(pre\_pred)|$ is the number of precondition predicates of an action of ith agent.
Mutual exclusion for actions:

$$\forall_{i=1}^{K}(\neg(a_{(i,x)}(t) \wedge a_{(i,y)}(t)))$$

where $x \neq y$, $\forall$ i x $\in A_i$ and y $\in A_i$
Consistency clauses:

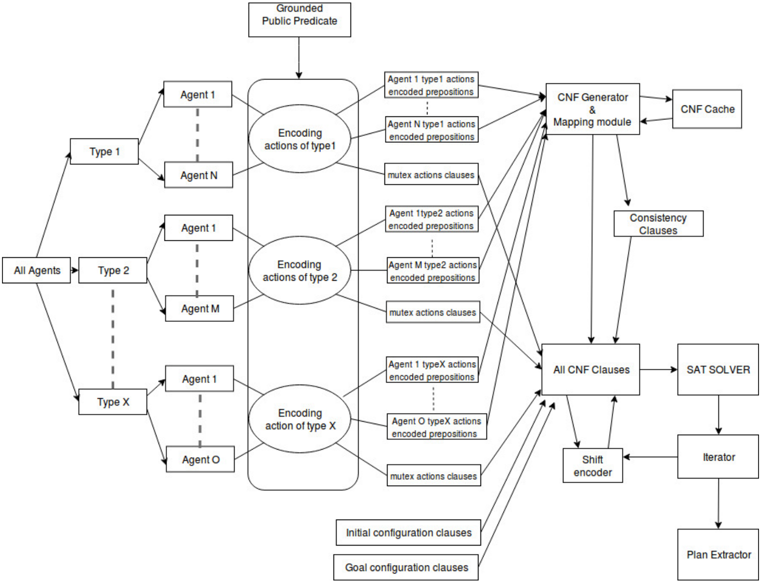
$$\forall_{i=1}^{|M|} \forall_{j=i+1}^{|M|} (\neg(Inst_{(i,t)} \wedge Inst_{(j,t)}))$$

Goal Condition:

$$Goal(T) = \bigwedge_{i=1}^{|X|} (x_{(i,T)} \models True)$$

Outline

Automated Planner Components: WAVE(our tool)



Outline

CoDMAP Planner Comparison

- CoDMAP planner Ranking

Planner	Optimality	Problem Coverage Ranking
ADP legacy*	NO	1
ADP*	NO	2
SIW+-then-BFS(f)	NO	3
CMAP_t	NO	4
DFS+	NO	5
Anytime-LAPKT	NO	6
CMAP_q	NO	7
MAPlan/FF+DTG	NO	8
PSM-VRD	NO	9
MADLA	NO	10
MAPR_p	NO	12
PMR	NO	11
PSM-VR	NO	13
MAPlan/LM-Cut*	YES	15
MAPlan/MA-LM-Cut*	YES	16
MARC	NO	17

Table: All the planners of the CoDMAP Competition

CoDMAP results

Problem	Mp Plan steps	Time (sec)	MpC Plan steps	Time (sec)	ADP-Legacy Steps	Time (sec)	ADP Steps	Time (sec)	WAVE Steps	Time (sec)
prob-9-0	70[25]	0.462	160[59]	3.787s	48[18]	0.23	42[16]	0.22	40[16]	3m5.291
prob-9-1	144[50]	0.616	198[68]	1.396s	42[16]	0.21	30[10*]	0.17	34[14]	25.541
prob-9-2	68[24]	0.189	156[54]	0.436s	32[14]	0.19	22[10]	0.17	30[14]	39.675
prob-10-0	212[71]	1.276	70[26]	5.187s	34[12]	0.2	56[22]	0.38	45[17]	4m0.956
prob-10-1	128[44]	1.032	250[89]	1.202s	42[18]	0.22	38[20]	0.22	45[17]	3m43.440
prob-10-2	86[29]	7.037	272[85]	18.32s	40[14]	0.21	34[12]	0.2	44[15]	2m39.221
prob-11-0	228[77]	8.064	208[65]	0.569s	34[12]	0.26	38[14]	0.24	42[14]	2m0.565
prob-11-1	196[61]	30.045	276[88]	1m10.8s	38[14]	0.24	50[16]	0.27	39[13]	3m2.994
prob-11-2	90[29]	0.562	128[48]	1.380s	58[22]	0.31	48[16]	0.27	52[17]	8m53.651
prob-12-0	126[41]	53.356	100[33]	40.29s	74[24]	0.45	56[20]	0.3	40[15]	8m51.995
prob-12-1	46[17]	29.127	378[119]	4m28.4s	42[12]	0.29	44[16]	0.28	40[16]	9m8.420
prob-13-0	TO	TO	TO	TO	42[12]	4.15	92[36]	0.61	TO	TO
prob-13-1	TO	TO	TO	TO	66[24]	0.36	66[28]	0.4	TO	TO
prob-14-0	132[44]	6m10.2	240[78]	2m24.5s	64[22]	0.44	42[28]	0.37	46[17]	22m54.18
prob-14-1	148[45]	2m54.0	TO	TO	46[16]	0.37	58[20]	0.36	45[15]	14m37.95
prob-15-0	214[73]	14m42.2	258[85]	7.194s	98[32]	0.8	82[32]	0.48	TO	TO
prob-15-1	TO	TO	TO	TO	78[30]	0.52	86[28]	0.61	TO	TO
prob-16-1	178[52]	30m15.9	TO	TO	74[30]	1.44	68[32]	0.70	TO	TO
prob-16-2	TO	TO	TO	TO	84[22]	1.76	74[24]	0.72	TO	TO
prob-17-1	TO	TO	TO	TO	138[42]	17.72	90[30]	1.60	TO	TO
Coverage	15	-	13	-	20	-	20	-	13	-

Table: Automated planner results on Blockworld domain

- Optimal total cost: 6/13 ADP , 7/13 ADP-legacy, 13/13 Mp and 13/13 MpC

CoDMAP results

Problem	Mp		MpC		ADP-legacy		ADP		Wave			WAVE
	Plan steps	Time (sec)	Plan steps	Time (sec)	Steps	Time (sec)	Steps	Time (sec)	Steps	total cost	Time (sec)	Optimal cost
pfile1	6[6]	0.004	6[6]	0.004s	6[6]	0.09	6[6]	0.07	6	6	5.789s	Yes
pfile2	16[8]	0.005	16[8]	0.004s	14[12]	0.08	14[12]	0.08	8	16	7.934s	-
pfile3	11[9]	0.004	11[9]	0.005s	11[11]	0.08	11[11]	0.08	7	14	7.192s	-
pfile4	13[9]	0.005	13[9]	0.005s	13[8]	0.09	13[8]	0.08	8	18	8.960s	-
pfile5	17[9]	0.006	17[8]	0.006s	25[19]	0.1	21[19]	0.09	10	22	13.244s	-
pfile6	10[6]	0.007	10[6]	0.007s	8[4]	0.1	8[4]	0.11	4	8	5.195s	Yes
pfile7	11[9]	0.01	11[9]	0.009s	12[12]	0.12	12[12]	0.11	7	18	9.238s	-
pfile8	28[11]	0.011	28[11]	0.014s	26[17]	0.12	26[17]	0.13	9	26	14.878s	Yes
pfile9	22[14]	0.012	23[14]	0.013s	23[16]	0.12	24[16]	0.12	10	20	19.218s	Yes
pfile10	17[8]	0.016	17[8]	0.013s	15[15]	0.16	15[15]	0.13	9	17	41.148s	-
pfile11	26[17]	0.019	23[13]	0.022s	20[11]	0.16	21[11]	0.15	11	21	1m24.8s	-
pfile12	39[20]	0.05	34[16]	0.042s	32[27]	0.21	32[26]	0.2	14	28	39m(TO)	-
pfile13	29[15]	0.055	28[11]	0.049s	23[21]	0.25	25[23]	0.23	13	25	9m30.4s	-
pfile14	43[17]	0.051	46[18]	0.056s	33[30]	0.28	32[29]	0.27	12	36	18m21.4s	-
pfile15	52[16]	0.156	44[16]	0.163s	41[26]	0.63	40[22]	0.61	ML	ML	ML	-
pfile16	67[18]	0.473	89[25]	0.484s	69[22]	3.68	66[26]	3.49	ML	ML	ML	-
pfile17	93[24]	0.724	102[25]	0.788s	80[35]	2.23	121[46]	4.96	ML	ML	ML	-
pfile18	133[34]	2.361	150[35]	2.255s	122[73]	8.86	122[68]	7.03	ML	ML	ML	-
pfile19	168[38]	2.347	172[45]	2.254s	144[93]	4m56.89	142[61]	4m3.87	ML	ML	ML	-
pfile20	191[33]	4.596	188[38]	4.784s	104[82]	8.06	116[95]	8.07	ML	ML	ML	-
Coverage	20	-	20	-	20	-	20	-	13	-	-	-

Table: Driver-log domain results

- Optimal total cost: 7/13 ADP , 6/13 ADP-legacy, 7/13 Mp and 10/13 MpC

CoDMAP results

Problem	MA-PLAN Total Cost	Total Time (sec)	MAPlan /MA-LM-Cut Total cost	Total Time (sec)	WAVE Total Cost	Total Time (sec)
probBLOCKS-9-0	20	1688.88	20	1781.12	40	316.577s
probBLOCKS-9-1	20	38.10	20	24.80	34	26.550s
probBLOCKS-9-2	-1	150.11	-1	250.97	39	63.07s
probBLOCKS-10-0	-1	1685.22	-1	1800.00	46	345.385s
probBLOCKS-10-1	-1	1686.14	-1	1780.24	42	381.875s
probBLOCKS-10-2	-1	1688.43	-1	1761.58	44	328.465s
probBLOCKS-11-0	-1	1690.02	-1	1800.02	39	207.308s
probBLOCKS-11-1	-1	1.00	-1	1.00	36	359.030s
probBLOCKS-11-2	-1	1687.06	-1	1800.00	45	744.841s
probBLOCKS-12-0	-1	1684.55	-1	1.00	41	401.532s
probBLOCKS-12-1	-1	1686.58	-1	1800.01	43	616.830s
probBLOCKS-13-0	-1	1687.30	-1	1800.01	65	TO
probBLOCKS-13-1	-1	1687.63	-1	1800.03	-1	TO
probBLOCKS-14-0	-1	1686.32	-1	1743.56	-1	TO
probBLOCKS-14-1	-1	1684.45	-1	1800.01	45	1575.589s
probBLOCKS-15-0	-1	1686.93	-1	1800.00	-1	TO
probBLOCKS-15-1	-1	1692.23	-1	1800.01	-1	TO
probBLOCKS-16-1	-1	1684.48	-1	1800.00	-1	TO
probBLOCKS-16-2	-1	1686.51	-1	1800.01	-1	TO
probBLOCKS-17-0	-1	1687.74	-1	1800.00	-1	TO
Coverage	2	-	2	-	12	-

Table: Total Cost comparison of MAPLAN and WAVE on Blockworld Domain

CoDMAP results

Problem	MAPLan /LM-Cut Total cost	Total Time (Sec)	MAPlan/ MA-LM-Cut Total cost	Total Time (Sec)	WAVE total cost	Total Time (Sec)
pfile1	**6**	6.15	**6**	8.69	**6**	5.579s
pfile2	13	1.66	-1	1.00	16	7.991s
pfile3	10	1.35	10	0.69	14	7.204s
pfile4	11	2.68	11	5.04	18	9.089s
pfile5	17	1.14	17	18.37	22	13.538s
pfile6	**8**	1.74	**8**	6.40	**8**	5.347s
pfile7	10	0.90	10	3.80	18	9.715s
pfile8	19	14.57	19	236.17	26	15.094s
pfile9	18	4.84	18	67.24	20	16.737s
pfile10	15	3.29	15	8.12	17	29.472s
pfile11	17	1.41	17	5.10	21	45.487s
pfile12	27	1219.88	-1	1.00	-1	1800.00s
pfile13	23	57.20	23	624.73	25	881.75s
pfile14	26	739.78	-1	1733.53	36	1424.083s
pfile15	29	1387.65	-1	1800.00	-1	ML
pfile16	-1	1.00	-1	1800.00	-1	ML
pfile17	-1	1694.16	-1	1.00	-1	ML
pfile18	-1	1714.66	-1	1800.00	-1	ML
pfile19	-1	1800.02	-1	1800.00	-1	ML
pfile20	-1	1800.00	-1	1800.00	-1	ML
Coverage	15	-	11	-	13	-

Table: Total Cost comparison of MAPLAN and WAVE on Driverlog Domain

- Optimal total cost: 2/13 optimal solution

CoDMAP results

Domain	Blockworld	Driverlog	Total	Rank
Planner	20	20	40	
ADP-legacy**	20	20	40	1
ADP**	20	20	40	2
PSM-VRD	20	20	40	3
MAPR-p	20	20	40	4
CMAP-t	20	19	39	5
DFS+	20	19	39	6
PMR	20	19	39	7
CMAP-q	19	19	38	8
SIW+-then-BFS(f)	20	17	37	9
MAPlan/FF+DTG	20	16	36	10
Anytime-LAPKT	20	15	35	11
Madagascar**	15	20	35	12
MADLA	19	15	34	13
PSM-VR	12	15	27	14
Wave**	12	13	25	15
MH-FMAP	0	17	17	16
MAPLan/LM-Cut**	2	13	15	17
MAPlan/MA-LM-Cut**	2	12	14	18
MARC	0	0	0	19

Table: CoDMAP problem coverage planner rank

Ma-sokoban results

Problem Name	Grid	(Agents, Boxes)	Plan Found	Plan Itera-tion	Total Time (Sec)
p01	7*7	2,2	YES	17	**169.11**
p01-1	7*7	2,2	YES	13	**127.19**
p02	12*6	2,3	T.O	-	-
p02-1	12*6	2,3	T.O	-	-
p03	7*7	2,3	YES	8	**93.9**
p03-1	7*7	2,3	YES	7	**117.0**
p04	7*11	3,4	T.O	-	-
p04-1	7*11	3,4	T.O	-	-
p05	10*9	3,4	T.O	-	-
p05-1	10*9	3,4	T.O	-	-
p06	10*9	2,4	YES	14	**644.1**
p06-1	10*9	3,4	T.O	-	-
p07	9*8	2,3	T.O	-	-
p07-1	9*8	3,3	T.O	-	-
p08	12*10	3,2	M.L.	-	-
p08-1	12*10	3,3	M.L.	-	-
p09	15*7	3,4	T.O	-	-
p09-1	15*7	3,4	T.O	-	-
p10	29*17	4,1	M.L.	-	-
p10-1	15*7	3,4	M.L.	-	-

Table: Sokoban Encoding Approach 1

Ma-sokoban Encoding Approach 1 planner comparison

Problem	Mp		MpC		ADP-legacy		ADP		Our approach (seq+PBL+py)	
	plan Steps	Time (sec)	plan Steps	Time (sec)	plan Steps	Time (sec)	plan Steps	Time (sec)	plan Steps	Time (sec)
p01	32(19)	0.026	32(19)	0.022	40(22)	0.120	34(21)	0.113	17	169.11
p01-1	20(14)	0.020	14(14)	0.020	28(15)	0.114	26(13)	0.102	13	127.19
p02	129(76)	0.550	80(74)	0.132	123(62)	0.39	133(74)	0.40	TO	TO
p02-1	159(92)	12.528	123(76)	1.106	145(74)	0.34	163(82)	0.31	TO	TO
p03	11(8)	0.032	11(8)	0.027	11(8)	0.14	11(8)	0.13	8	93.947
p03-1	10(8)	0.018	10(8)	0.023	17(10)	0.12	17(10)	0.13	7	117.080
p04	65(33)	0.270	71(31)	0.295	102(38)	1.25	134(55)	1.48	TO	TO
p04-1	TO	30M	TO	30M	TO	TO	TO	TO	TO	TO
p05	85(41)	0.434	65(27)	0.225	54(51)	1.98	56(53)	1.90	TO	TO
p05-1	24(21)	0.240	70(26)	0.258	24(21)	0.31	42(21)	0.36	TO	TO
p06	25(14)	0.140	25(14)	0.155	30(16)	0.41	37(22)	0.45	14	644.169
p06-1	42(24)	0.403	41(16)	0.218	29(14)	0.88	29(14)	0.89	TO	TO
p07	73(49)	0.602	63(39)	0.103	85(43)	0.50	87(45)	0.51	TO	TO
p07-1	44(24)	0.122	52(29)	0.135	40(26)	0.25	38(24)	0.40	TO	TO
p08	157(90)	3.936	169(92)	1m3.45	168(96)	01.73	172(96)	1.42	M.L.	M.L.
p08-1	126(56)	9.043	129(49)	2m19.5	124(85)	0.89	124(85)	0.98	M.L.	M.L.
p09	91(42)	0.606	120(54)	0.455	176(78)	4m57.83	111(52)	3m53.53	TO	TO
p09-1	TO	30M	TO	30M	TO	TO	TO	TO	TO	TO
p10	TO	30M	296(223)	4m24.0	216(211)	04.87	216(211)	4.81	M.L.	M.L.
p10-1	73(36)	0.560	73(38)	0.433	171(68)	15m27.8	149(57)	15m42.6	M.L.	M.L.
Coverage	17	-	18	-	18	-	18	-	5	-

Table: All Planner comparison for Encoding Approach 1

Ma-sokoban results

Problem	Grid Size N*M	Agents	Boxes	ENCODING 1(seq+PBL +python) Make-Span	Time(sec)	ENCODING 1(seq +python) Make-Span	Time(sec)
p01	7*7	2	2	17	169.11	17	10.227
p01-1	7*7	2	2	13	127.19	13	5.66
p03	7*7	2	3	8	93.947	8	5.53
p03-1	7*7	2	3	7	117.08	7	6.35

Table: PBL removal on Encoding Scheme 1 result

Problem	Grid	Robot	Box	Expo._jump Time (Sec)	ADP Time (Sec)	Mp Time (Sec)	Mpc Time (Sec)	M Time (Sec)	Black BOX Time (Sec)	SAT-PLAN Time (Sec)
Test_1	7*7	2	2	0.043s [13]	0.131 [13]	0.019**	0.018	0.033	3.91	INC. ***
Test_2	15*15	2	2	1.065s [18]	1.722 [25]	7.941s [19]	7.867 [19]	8.760 [20]	11.68hrs	INC.
Test_3	30*30	2	2	18.957s [29]	14.56 [29]	11m29.9s [29]	11m53s [29]	11m20s [29]	T.O.	INC.

Table: ALL Planner comparison with exponential jumps

Notation:
1. O.S. is optimal steps (makespan) for solving the problem.
2. *Time is in seconds (user+sys mode of process execution)
3. ** Min time taken when multiple run gives different values.
4 .***INC means that conversion from Ma-pddl to pddl is not compatible with the planner.

Test Cases: There are three test cases the configuration is given below.
Test_1: CoDMAP sokoban problem 2
Test_2: A1:(0,0) A2:(14,0)[Agents] B1:(1,1)B2:(13,1)[Boxes] G1:(14,14) G2:(0,14) [Goals]
Test_3: A1:(0,0) A2:(29,0)[Agents] B1:(1,1)B2:(28,1)[Boxes] G1:(29,29)G2:(0,29) [Goals]

Ma-sokoban results: : Neighborhood Encoding Approach

Problem	Grid	Agent	Box	Total Time	Clause Generation Time	SAT solver Time	Steps	ADP (Time)	ADP (Make Span)
p01	7*7	2	2	10.227	4.808	5.419	17	0.020	22
p01-1	7*7	2	2	5.66	3.7331	1.928	13	0.005	13

Table: Encoding scheme 1 result

Problem	Grid	Agent	Box	Total Time	Clause Generation Time	SAT solver Time	Steps	ADP (Time)	ADP (Make Span)
p01	7*7	2	2	**1.94083**	1.27823	0.662594	17	0.020	22
p01-1	7*7	2	2	**0.97435**	0.67516	0.29937	13	0.005	13

Table: Neighborhood Encoding Approach with encoding scheme 2 result

Ma-sokoban results : Relaxed jump with neighborhood Approach

Problem	Mp		MpC		ADP-legacy		ADP		Our approach (seq+NEG+Rel_jmp)	
	plan Steps	Time (sec)	plan Steps	Time (sec)	plan Steps	Time (sec)	plan Steps	Time (sec)	plan Steps	Time (sec)
p01	32(19)	0.026	32(19)	0.022	40(22)	0.120	34(21)	0.113	17	0.159
p01-1	20(14)	0.020	14(14)	0.020	28(15)	0.114	26(13)	0.102	13	0.072
p02	129(76)	0.550	80(74)	0.132	123(62)	0.39	133(74)	0.40	51	2m49.56
p02-1	159(92)	12.528	123(76)	1.106	145(74)	0.34	163(82)	0.31	45	2m24.76
p03	10(8)	0.032	10	0.027	11	0.14	11	0.13	8	0.038
p03-1	10(8)	0.018	10(8)	0.023	17(10)	0.12	17(10)	0.13	7	0.033
p04	65(33)	0.270	71(31)	0.295	102(38)	1.25	134(55)	1.48	21	6m47.69
p04-1	TO	30M	TO	30M	TO	TO	TO	TO	TO	TO
p05	85(41)	0.434	65(27)	0.225	54(51)	1.98	56(53)	1.90	20	4.279
p05-1	24(21)	0.240	70(26)	0.258	24(21)	0.31	42(21)	0.36	16	1.204
p06	25(14)	0.140	25(14)	0.155	30(16)	0.41	37(22)	0.45	14	0.352
p06-1	42(24)	0.403	41(16)	0.218	29(14)	0.88	29(14)	0.89	12	0.310
p07	73(49)	0.602	63(39)	0.103	85(43)	0.50	87(45)	0.51	31	39.574
p07-1	44(24)	0.122	52(29)	0.135	40(26)	0.25	38(24)	0.40	19	0.991
p08	157(90)	3.936	169(92)	1m3.45	168(96)	01.73	172(96)	1.42	TO	TO
p08-1	126(56)	9.043	129(49)	2m19.5	124(85)	0.89	124(85)	0.98	29	27.01
p09	91(42)	0.606	120(54)	0.455	176(78)	4:57.83	111(52)	3:53.53	31	22.72
p09-1	TO	30M	TO	30M	TO	TO	TO	TO	TO	TO
p10	TO	30M	296(223)	4m24.0	216(211)	04.87	216(211)	4.81	TO	TO
p10-1	73(36)	0.560	73(38)	0.433	171(68)	15:27.8	149(57)	15:42.6	28	40.94
Coverage	17	-	18	-	18	-	18	-	16	-

Table: All Planner comparison with relaxed jump with neighborhood encoding approach

Outline

Limitation, conclusion and Future Work

Limitations:

- The main reason of tool's slow performance is slow performance of the SAT solver.
- Our planner does not give optimal total cost of the problem.
- Our planner does not supports features of MA-PDDL which is not highlighted in RED.

Future work

- Introduce cardinality constraints for plan optimality in terms of total cost.
- Introduce Incremental approach in the SAT solver to optimize the SAT solver invocation time.
- Introduce the plan validator module in the tool.
- New features of MA-PDDL can be added in the tool.
- Relaxed Jump using graph approach.
- Scalability.

QA

- Questions ?